TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

DIPLOMOVÁ PRÁCE

Liberec 2011 **Kamil Polák**

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Realizace digitálních audio efektů na signálovém procesoru TMS320C6416

Design of Digital Audio Effects on TMS320C6416

Diplomová práce

Autor: **Bc. Kamil Polák**
Vedoucí práce: Ing. Zbyněk Koldovský, Ph.D.
Konzultant: Ing. Miroslav Holada, Ph.D.

V Liberci 20. 5. 2011

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Akademický rok: 2010/11

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	Bc. Kamil Polák
Osobní číslo:	M08000225
Studijní program:	N2612 Elektrotechnika a informatika
Studijní obor:	Informační technologie
Název tématu:	Realizace digitálních audio efektů na signálovém procesoru TMS320C6416
Zadávající katedra:	Ústav informačních technologií a elektroniky

!ZDE VLOŽIT ORIGINÁLNÍ ZADÁNÍ!

Zásady pro vypracování:

1. Seznámení s principy vybraných typů digitálních audio efektů (ekvalizér, delay, chorus, vibrato, wah-wah, phaser, nelineární efekty, atp.).
2. Seznámení s procesorem TMS320C6416, s vývojovým prostředím a programovacím jazykem pro procesor, realizace základního zapojení a nastavení pro on-line zpracování.
3. Realizace vybraných efektů v procesoru a vytvoření nástroje pro ovládání parametrů, obojí v reálném čase.
4. Výsledek práce předveďte při obhajobě DP.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah průvodní zprávy:	cca 40-50 stran
Forma zpracování diplomové práce:	tištěná / elektronická

Seznam odborné literatury:

[1] U. Zölzer et al.: DAFX: Digital Audio Effects, Wiley, ISBN-13: 978-0471490784, 2002.

[2] Rulph Chassaing: Digital Signal Processing and Applications with the TMS320C6713 and TMS320C6416 DSK, Wiley-IEEE Press, ISBN-13: 978-0470138663, 2008.

Vedoucí diplomové práce:	**Ing. Zbyněk Koldovský, Ph.D.** Ústav informačních technologií a elektroniky
Konzultant diplomové práce:	**Ing. Miroslav Holada, Ph.D** Ústav informačních technologií a elektroniky
Datum zadání diplomové práce:	**1. října 2010**
Datum odevzdání diplomové práce:	**20. května 2011**

prof. Ing. Václav Kopecký, CSc.
děkan

doc. Ing. Zdeněk Plíva, Ph.D.
pověřen vedením ústavu

V Liberci dne 1. října 2010

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména §60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé diplomové práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.)

Jsem si vědom, že užití své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum 20. 5. 2011

Podpis

Poděkování

Zde bych rád poděkoval vedoucímu své práce panu Ing. Koldovskému, Ph.D. za cenné rady a pomoc při konzultacích a dále konzultantovi panu Ing. Holadovi, Ph. D. za podporu a zapůjčení vývojových prostředků.

Abstract

The goal of this thesis is to describe the main fundamental of chosen digital audio effects and implementation with the Starter Kit DSK C6416 containing digital signal processor (DSP) Texas Instruments TMS320C6416. The main part of this work was to design digital audio effects in the C programming language using the Code Composer Studio IDE. The analogue low frequency input signal is converted via AD converter on the DSK board and then the algorithms of real-time digital signal processing follow. After processing on DSP, the digital signal is converted back to the analog domain via DA converter. This work deals with delay audio effects or delay-based effects (chorus, vibrato, flanger), effects applying digital filters with constant or variable coefficients (equalizer, wah-wah, phaser), and other nonlinear effects (distortion, valve simulator). Some stereo enhancers are addressed as well. The next part of the work describes the graphic user interface application running on a PC that allows to control the DSP through USB using the RTDX technology. This application was written in the MATLAB programming language.

Keywords: Digital Signal Processor (DSP), TMS320C6416, digital audio effects, MATLAB, C, RTDX

Abstrakt

Cílem této práce je seznámení s principy digitálních audio efektů a dále jejich realizace s využitím vývojové desky DSK C6416 osazené digitálním signálovým procesorem (DSP) Texas Instruments TMS320C6416. Hlavním úkolem bylo naprogramovat algoritmy realizující zvukové efekty v programovacím jazyku C s využitím vývojového prostředí Code Composer Studio. Použitá vývojová deska je vybavena rovněž AD a DA převodníky. Analogový nízkofrekvenční signál je po digitalizaci zpracován v reálném čase algoritmem běžícím na DSP a následně zpět převeden do analogové podoby. V textu práce lze nalézt teoretický základ, návrh a implementaci efektů pracujících na principu konstantního a variabilního zpoždění zvukových vzorků (delay, chorus, vibrato, flanger), dále efekty realizované pomocí číslicových filtrů s konstantními a proměnnými koeficienty (ekvalizér, wah-wah, phaser). Další část je věnována nelineárním efektům (zkreslení, simulace elektronkového zesilovače) a stereofonním korektorům. Jako součást práce byla vytvořena rovněž grafická aplikace běžící na PC, která umožňuje ovládání efektů v DSP. Aplikace byla realizována ve skriptovacím jazyku MATLAB a s vývojovou deskou komunikuje po sběrnicí USB pomocí technologie RTDX.

Klíčová slova: Digitální signálový procesor (DSP), TMS320C6416, digitální audio efekty, MATLAB, C, RTDX

Obsah

PROHLÁŠENÍ 3

PODĚKOVÁNÍ 4

ABSTRACT 5

ABSTRAKT 6

OBSAH 7

SEZNAM OBRÁZKŮ A TABULEK 9

SEZNAM ZKRATEK A SYMBOLŮ 11

1 ÚVOD 13

1.1 VYUŽITÍ AUDIO EFEKTŮ 14
1.2 MINIMUM Z HUDEBNÍ TEORIE 15
1.2.1 Výška tónu 16
1.2.2 Barva tónu 16
1.2.3 Intenzita tónu 16
1.2.4 Skládání tónů 17

2 PRINCIPY VYBRANÝCH AUDIO EFEKTŮ 18

2.1 TREMOLO 18
2.2 DELAY, REVERB 19
2.3 EFEKTY VYUŽÍVAJÍCÍ VARIABILNÍHO ZPOŽDĚNÍ SIGNÁLU 21
2.3.1 Vibráto 22
2.3.2 Flanger, chorus 22
2.4 EFEKTY REALIZOVANÉ POMOCÍ FILTRŮ 23
2.4.1 Allpass filtr 23
2.4.2 Ekvalizér 25
2.5 EFEKTY REALIZOVANÉ POMOCÍ LADITELNÝCH FILTRŮ 26
2.5.1 Wah-wah 26
2.5.2 Phaser 27
2.5.3 Variabilní dolní propust 27
2.6 NELINEÁRNÍ AUDIO EFEKTY 27
2.6.1 Zkreslení 28
2.6.2 Simulace elektronkového zesilovače 30
2.6.3 Vliv aliasingu na efekty zkreslení 31
2.7 STEREOFONNÍ EFEKTY 32

3 POUŽITÉ HW A SW VÝVOJOVÉ PROSTŘEDKY 35

3.1 DIGITÁLNÍ SIGNÁLOVÝ PROCESOR TMS320C6416 35
3.2 VÝVOJOVÝ DESKA C6416 DSK 38
3.2.1 AD a DA převodníky 39
3.3 CODE COMPOSER STUDIO 39
3.3.1 Práce v prostředí CCS 41
3.3.2 Vytvoření nového projektu v CCS 42
3.3.3 Ladění programu pro DSP v CCS 42
3.3.4 Systém DSP/BIOS 43
3.3.5 Systém RTDX 44

4 NÁVRH A REALIZACE ALGORITMŮ VÝSLEDNÝCH EFEKTŮ 45

4.1 JEDNOTKA PRO ČASOVĚ MODULOVANÉ EFEKTY 46
4.1.1 Implementace neceločíselného zpoždění vzorků 49
4.1.2 Návrh číslicové dolní propusti s IIR filtrem 2. řádu 49
4.1.3 Implementace IIR filtru v signálovém procesoru 53
4.2 STEREOFONNÍ EFEKTY 54
4.2.1 Stereo chorus, vibráto 54

4.2.2 Rozšíření stereofonní báze 55
4.2.3 Stereo reverb 56
4.2.4 Vytvoření prostorového efektu u mono signálů 58
4.3 EFEKTY REALIZOVANÉ POMOCÍ LADITELNÝCH FILTRŮ 60
4.3.1 Wah-wah 60
4.3.2 Phaser 61
4.3.3 Variabilní dolní propust 63
4.4 JEDNOTKA PRO REALIZACI EFEKTŮ ZKRESLENÍ 63
4.4.1 Vliv dolní propusti na charakter zvuku zkreslené kytary 64
4.4.2 Struktura jednotky realizující efekty zkreslení 66
4.4.3 Implementace IIR filtrů vyšších řádů 67
4.5 EKVALIZÉR A OVLÁDÁNÍ HLASITOSTI. 68
4.5.1 Ovládání hlasitosti 69
4.6 SIGNALIZACE ÚROVNĚ SIGNÁLU 69

5 REALIZACE OVLÁDÁNÍ EFEKTŮ A KOMUNIKACE DSP S PC 71

5.1 GRAFICKÉ UŽIVATELSKÉ ROZHRANÍ PRO OVLÁDÁNÍ EFEKTŮ 71
5.2 IMPLEMENTACE RTDX KOMUNIKACE V MATLABU 73
5.2.1 Inicializace RTDX komunikace 73
5.2.2 Odesílání dat RTDX kanálem 75
5.3 IMPLEMENTACE RTDX KOMUNIKACE NA DSP 75

6 CELKOVÉ SCHÉMA PROGRAMU REALIZOVANÝCH EFEKTŮ 77

7 ZHODNOCENÍ VÝSLEDKŮ A ZÁVĚR 78

SEZNAM POUŽITÉ LITERATURY 80

PŘÍLOHA A – POPIS OBSAHU PŘILOŽENÉHO CD 82

PŘÍLOHA B – UKÁZKA ZDROJOVÉHO KÓDU VYBRANÝCH EFEKTŮ 83

PŘÍLOHA C – VÝZNAM DAT PŘI KOMUNIKACI MEZI PC A DSP 84

Seznam obrázků a tabulek

Obrázky:

Obr. 1.1: Praktická část práce a použité prostředky 14
Obr. 2.1: principiální schéma efektu tremolo 18
Obr. 2.2: Tremolo – amplitudová modulace signálu 19
Obr. 2.3: Základní struktura delay efektu – hřebenový FIR filtr 19
Obr. 2.4: Amplitudová charakteristika hřebenového FIR filtru 20
Obr. 2.5: Struktura delay efektu s hřebenovým IIR filtrem 20
Obr. 2.6: Amplitudová charakteristika hřebenového IIR filtru 21
Obr. 2.7: Impulsní odezva hřebenového IIR filtru 21
Obr. 2.8: Princip efektu vibráto 22
Obr. 2.9: Základní struktura efektů chorus, flanger, (vibráto) 22
Obr. 2.10: Fázová charakteristika allpass filtru 1. řádu pro $f_c = 0{,}01fs$ 24
Obr. 2.11: Realizace filtrů LP, HP, BP, BR pomocí allpass fitru 24
Obr. 2.12: Sériová struktura ekvalizéru 26
Obr. 2.13: Princip efektu wah-wah 26
Obr. 2.14: Princip efektu phaser 27
Obr. 2.15: Variabilní dolní propust 27
Obr. 2.16: Příklad průběhu nelineární funkce sinus hyperbolický 28
Obr. 2.17: Symetrická limitace harmonického signálu (*distortion*) 29
Obr. 2.18: Asymetrická limitace harmonického signálu (*fuzz*) 30
Obr. 2.19: Aproximace VA char. elektronkového zesilovače kvadratickou funkcí 30
Obr. 2.20: Simulace elektronkového zesilovače 31
Obr. 2.21: Vliv aliasingu na efekt zkreslení při nízké vzorkovací frekvenci (1,9 kHz) 32
Obr. 2.22: Stereofonní poslech pomocí sluchátek 33
Obr. 2.23: Časová diference mezi levým a pravým kanálem signálu 34
Obr. 3.1: Blokové schéma jádra procesoru TMS320C64xx 37
Obr. 3.2: Vývojová deska C6416 DSK 38
Obr. 3.3: Blokové schéma vývojové desky C6416 DSK [5] 39
Obr. 3.4. Popis tvorby programu v prostředí Code Composer Studio 40
Obr. 3.5. Tvorba programu pro DSP 40
Obr. 3.6. Ukázka prostředí Code Composer Studio 3.1 41
Obr. 3.7. Konfigurace systému DSP/BIOS 43
Obr. 4.1: Struktura realizující základní zpožďovací efekty 46
Obr. 4.2: Shrnutí funkcí základní efektové jednotky 47
Obr. 4.3: Struktura realizující časově modulované efekty 47
Obr. 4.4: Princip interpolace neceločíselného zpoždění vzorků signálu 49
Obr. 4.5: Zobrazení pólů přenosu Butterworthova LP filtru 2. řádu v *s*-rovině. 51
Obr. 4.6: Frekvenční char. Butterworthova LP filtru pro $f_0 = 100$ Hz a $f_S = 48$ kHz 53
Obr. 4.7: *Přímá struktura I* filtru IIR 2. řádu 53
Obr. 4.8: *Přímá kanonická struktura II* filtru IIR 2. řádu 54
Obr. 4.9: Rozdíl stereofonního poslechu pomocí sluchátek a reproduktorů 55
Obr. 4.10: Korektor šířky stereofonní báze 55
Obr. 4.11: Prostorový efekt využívající zpoždění signálu 56
Obr. 4.12: Prostorový efekt využívající zpoždění signálu a zpětné vazby 57
Obr. 4.13: Prostorový efekt u mono signálů 58
Obr. 4.14: Frekvenční char. dvou hřebenových FIR filtrů vytvářející stereo ozvěnu 59
Obr. 4.15: Prostorový efekt u mono signálu využívající zpoždění signálu a zpětné vazby 59
Obr. 4.16: Realizace efektu wah-wah s BP filtrem sestaveným z allpass filtru 2. řádu 60
Obr. 4.17: Spektrogram bílého šumu zpracovaného efektem wah-wah 61
Obr. 4.18: Realizace efektu phaser pomocí dvojice přeladitelných pásmovou propustí 62
Obr. 4.19: Spektrogram bílého šumu zpracovaného efektem phaser 62
Obr. 4.20: Variabilní dolní propust 63
Obr. 4.21: Spektrogram bílého šumu zpracovaného variabilním LP filtrem 63
Obr. 4.22: Vícepásmové zpracování signálu 64
Obr. 4.23: Frekvenční průběh čisté a zkreslené kytary, struna d (147 Hz) 65

Obr. 4.24: Frekvenční průběh čisté a zkreslené kytary upravené LP filtrem, struna d (147 Hz) 65
Obr. 4.25: Jednotka realizující kytarové efekty zkreslení 66
Obr. 4.26: Přímá a kaskádní struktura filtru IIR 67
Obr. 4.27: Realizace 4-pásmového ekvalizéru 68
Obr. 4.28: Amplitudové charakteristiky ekvalizačních filtrů 69
Obr. 5.1: Grafické uživatelské rozhraní ovládacího programu 71
Obr. 5.2: Dialog pro volbu cíle pro RTDX komunikaci 73
Obr. 6.1: Celkové blokové schéma realizovaných digitálních audio efektů 77

Tabulky:

Tabulka 4.1: Souhrn funkcí jednotky realizující základní zpožďovací efekty: 46
Tabulka 4.2: Příklady audio efektů v závislosti na parametrech: 48
Tabulka 4.3: Vliv hodnoty zpoždění na efekt stereo korektoru: 57
Tabulka 4.4: Vliv výstupních parametrů na zvukový charakter efektu: 67
Tabulka 4.5: LED indikátor úrovně signálu: 70

Seznam zkratek a symbolů

Allpass filtr

Filtr ovlivňující pouze fázový posun signálu, nikoliv amplitudu

ALU (Arithmetic Logic Unit)

Aaritmeticko-logická jednotka

BP (Band-Pass)

Pásmová propust (typ filtru)

BR (Band-Reject)

Pásmová zádrž (typ filtru)

CPLD (Complex Programmable Logic Device)

Programovatelný logický obvod

dB (decibel)

Logaritmická jednotka

DMA (Direct Memory Access)

Přímý přístup do paměti

DSP (Digital Signal Procesor)

Digitální signálový procesor

EMIF (External Memory Interface)

Rozhraní pro připojení vnější paměti

FFT (Fast Fourier transform)

Rychlá Fourierova transformace

FIR (Finite Impulse Response)

Konečná impulsní odezva

GEL (General Extension Language)

Jazyk rozšiřující vývojové prostředí CCS

GUI (Graphical User Interface)

Grafické uživatelské rozhraní

HP (High-Pass)

Horní propust (typ filtru)

I/O (Input/Output)

Vstupy a výstupy

IIR (Infinite Impulse Response)

Nekonečná impulsní odezva

JTAG (Joint Test Action Group)

Standard pro emulaci, testování a ladění el. obvodů na čipu

LP (Low-Pass)

Dolní propust (typ filtru)

LTI system (Linear Time-Invariant system)

Lineární časově invariantní systém

McBSP (Multichannel Buffered Serial Port)

Vícekanálový sériový port

MIPS (Million Instructions Per Second)

Měřítko pro porovnávání teoretické rychlosti procesorů

NF technika

Nízko-frekvenční technika

Pipelining

Zřetězené zpracování dat – technika paralelního zpracování

RTDX (Real Time Data eXchange)

Technologie (systém) pro přenos dat v reálném čase vyvinutý společností Texas Instruments využívaný zejména pro ladění programu v DSP

SIMD (Single Instruction, Multiple Data)

Obecný paralelní systém, který je schopný pomocí jednoho algoritmu zpracovávat několik různých dat zároveň

TI (Texas Instruments)

Společnost, výrobce el. integrovaných obvodů

VA charakteristika

Volt-ampérová charakteristika

VLIW (Very Long Instruction Wold)

Architektura (signálových) procesorů

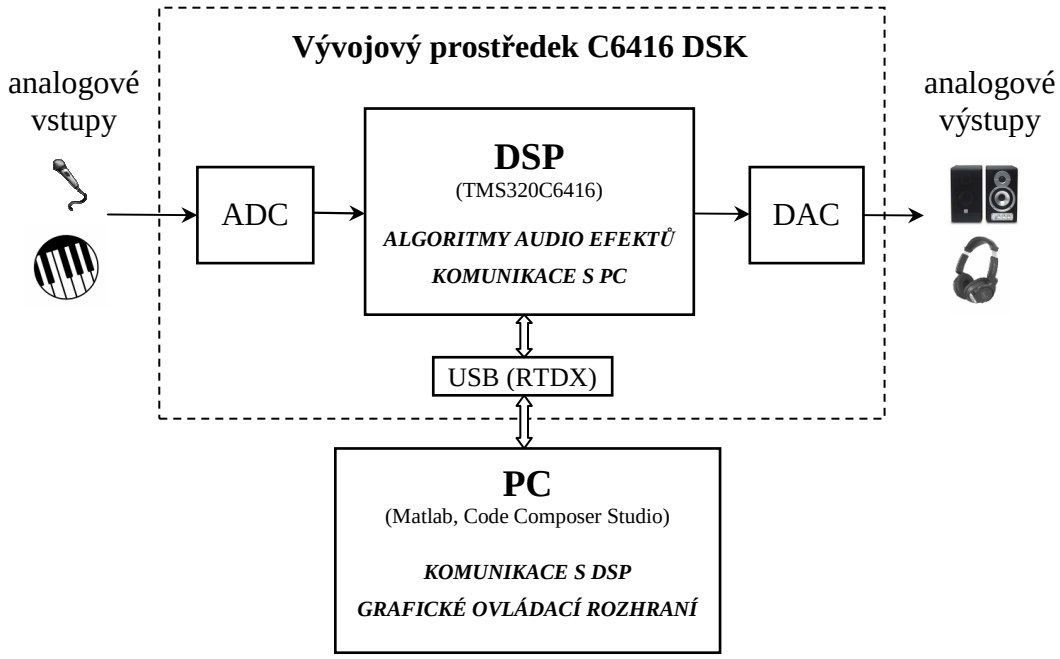
1 Úvod

Rozvoj číslicové elektroniky přinesl koncem 20. století velké množství nových technických prostředků do většiny oblastí lidských činností. První signálové procesory, které se objevily na přechodu sedmdesátých a osmdesátých let minulého století, nabídly do té doby nevídané možnosti pro zpracování číslicových signálů. Digitální audio efekty, které lze zařadit do oblasti číslicového zpracování akustických signálů, také ve většině případů díky lepším možnostem a nižší ceně postupně nahradily efekty vytvářené pomocí analogových elektrických obvodů případně ještě starší elektromechanické přístroje. Jako příklad lze uvést jednoduchý zvukový efekt, kterým je zpoždění případně ozvěna. V dobách před nástupem digitální techniky se podobné efekty tvořily velmi obtížně, např. pomocí elektromagnetické pásky rotující v kruhu vybavené zapisovací a čtecí hlavou. Avšak zvuková kvalita takových zařízení zejména s dobou provozu značně klesala. V současnosti lze zpožďovací efekty vytvořit naprosto jednoduše a elegantně pomocí číslicových obvodů a pamětí. Cílem této práce je tedy shrnout principy základních typů digitálních audio efektů a využít možnosti digitálního signálového procesoru (DSP) pro jejich realizaci.

Samotný text práce je rozdělen do několika kapitol. V první kapitole je naznačen cíl práce, jsou zde uvedeny obecné příklady použití audio efektů a rovněž vysvětleny některé základní pojmy z hudební teorie. Kapitola druhá se zabývá obecnými principy vybraných digitálních efektů. Ve třetí kapitole jsou blíže popsány použité hardwarové prostředky, zejména vývojová deska a signálový procesor. Přibližně stejný prostor je rovněž věnován popisu použitého programového vybavení a softwarových technologií. Kapitola čtvrtá se již zabývá návrhem a vlastní implementací zvukových efektů. Pátá kapitola je věnována návrhu grafické aplikace pro ovládání efektů, popsána je také komunikace mezi DSP a PC. V rámci šesté kapitoly lze nahlédnout na blokové schéma kompletního programu realizující audio efekty. Poslední, sedmá kapitola tvoří závěrečné zhodnocení celé práce, subjektivní posouzení kvality dosažených efektů, porovnání s komerčními produkty a dále diskusi možných vylepšení. Zmíněny jsou rovněž některé problémy, které se objevily při implementaci algoritmů a jiných částí programu.

Hlavní cíl a náplň práce shrnuje obrázek 1.1. Jednotlivé efekty jsou realizované s využitím signálového procesoru TMS320C6416, který zpracovává vzorky z AD převodníku a upravené je posílá na výstup. Pro ovládání efektů slouží externí grafická

aplikace, která komunikuje s DSP za běhu programu sběrnicí USB. Vidíme zde rovněž možné analogové vstupy a výstupy a použité vývojové prostředky.



Obr. 1.1: Praktická část práce a použité prostředky

1.1 Využití audio efektů

Uplatnění audio efektů nalezneme zejména v hudebním průmyslu, kde mohou sloužit pro úpravu zvukového projevu analogových hudebních nástrojů. Příkladem této kategorie jsou efekty využívané při hře na klasickou či elektrickou kytaru. Díky variabilním úpravám zvuku nalézá tento nástroj uplatnění v širokém spektru hudebních žánrů. Jiným příkladem mohou být již čistě elektronické hudební nástroje, jako např. el. klávesy. Podobné přístroje umožňují kromě vlastní syntézy tónů také dodatečnou úpravu výsledného zvuku pomocí zvukových efektů. Zvolený nástroj lze obohatit např. přidáním ozvěny, či zvýrazněním některých frekvenčních pásem pomocí ekvalizéru. V obou předchozích případech je kladen důraz na zpracování zvukového signálu v reálném čase s co nejmenším zpožděním, jelikož delší odezva působí při hře na hudební nástroj pro hudebníka velmi rušivě. Stejné nároky platí ve většině případů v oblasti ozvučovací techniky. Mixážní pulty musí být schopny slučovat a upravovat velké množství signálů v reálném čase. Naopak opačnou kategorií jsou počítačové programy pro úpravu již nahraných záznamů, které naleznou uplatnění např. při editaci hudebních nahrávek nebo při úpravě audio stopy u video nahrávek.

Z jiné oblasti lze uvést např. zpracování lidské řeči. Správně nastavený ekvalizér může zvýšit srozumitelnost mluveného slova v daném prostředí. Stejného principu využívají i lidská naslouchátka. Lékař se snaží přibližně zjistit frekvenční amplitudovou charakteristiku poškozeného orgánu tak, že otestuje práh citlivosti na zvuky o různých frekvencích. V naslouchadle poté nastaví filtry tak, aby se výsledná křivka co nejvíce podobala zdravému orgánu. V některých případech se naopak snažíme mluvený projev změnit, abychom nebyli schopni identifikovat původního řečníka, čehož se využívá např. při zveřejnění výpovědi svědků v médiích.

Jako poslední příklad lze uvést spotřební elektroniku. Téměř každý CD/MP3 přehrávač umožňuje základní úpravu zvuku. Rovněž televizory bývají často vybaveny ekvalizérem pro úpravu frekvenčního spektra zvuku. První televizory byly vybaveny čistě analogovými korekčními obvody, tvořené z diskrétních součástek, které uživatel ladil pomocí potenciometrů. S rozšířením integrované elektroniky a příchodem dálkového ovládání nahradily diskrétní obvody speciální el. integrované obvody. Ovládání těchto obvodů bylo realizováno po sběrnicích pomocí číslicové elektroniky, avšak samotné zpracování signálu zůstávalo analogové. Současné televizory, které jsou již schopny zpracovávat signály z digitálního vysílaní, jsou však již běžně vybaveny ekvalizéry, případně efekty imitujícími prostorový zvuk, realizovanými pomocí digitálních signálových procesorů (DSP) či jiných číslicových obvodů. Složitější škálou efektů jsou poté vybaveny např. AV receivery, které tvoří základní komponentu domácího kina.

1.2 Minimum z hudební teorie

Ačkoli je tato práce psaná zejména z technického pohledu, pro správné pochopení principů některých efektů je vhodné znát několik základních pojmů z oblasti hudební teorie.

Základní stavební prvek hudby se nazývá tón. Tón lze popsat jako periodický akustický signál, jehož základní frekvence se nemění. Ke tvorbě tónů mohou sloužit hudební nástroje případně lidské ústrojí. V hudbě se rovněž používají i neperiodické zvuky, které vznikají u hudebních nástrojů jako např. činely nebo bubny. Každý tón lze popsat pomocí následující vlastností: *výška*, *barva*, *intenzita*, případně délka.

1.2.1 Výška tónu

Absolutní výška tónu určuje frekvenci základní (nejnižší) harmonické složky konkrétního tónu. Jako výchozí hodnota pro ladění hudebních nástrojů se udává frekvence 440 Hz, jedná se o tzv. komorní *a*. Z hlediska lidského vnímání frekvencí zvuku má daleko větší význam relativní výška tónu, tedy poměr dvou tónů o rozdílných frekvencích f_1 a f_2 vůči sobě. V hudbě se tento poměr $f_2 : f_1$, kde $f_2 > f_1$, nazývá interval. Důležité jsou zejména intervaly, jejichž frekvence lze vyjádřit poměrem malých celých čísel. Interval, jehož tóny mají kmitočty v poměru 2:1, se nazývá oktáva. Tón, který je na stupnici o oktávu výše, má tedy oproti původnímu tónu dvojnásobnou frekvenci. Další významné intervaly, které určují rozdělení oktávy, jsou uvedeny např. zde [6].

1.2.2 Barva tónu

Tóny produkované hudebními nástroji jsou zpravidla periodické a složené z několika harmonických složek. Po provedení frekvenční analýzy lze kromě základní frekvence (první harmonické) pozorovat i další frekvenční složky, jejichž frekvence je rovna celočíselnému násobku první harmonické, tzv. vyšší harmonické. Barva tónu je daná počtem a amplitudou vyšších harmonických složek. Tóny, které obsahují větší počet vyšších harmonických s větší amplitudou, znějí ostřeji než tóny, kde nejsou vyšší harmonické kmitočty významně zastoupeny. Tohoto principu se využívá např. u kytarových efektů, kde se úmyslným zavedením zkreslení signálu zvyšuje počet vyšších harmonických složek.

1.2.3 Intenzita tónu

Intenzita tónu je dána jeho amplitudou. Při korekci hlasitosti je však nutné zohlednit nelineární vlastnosti lidského sluchu, používají se proto logaritmické stupnice udávané v dB. Pro hladinu akustického tlaku v dB platí následující vztah [8]:

$$L_p = 20\log(p/p_0)\ \text{[dB; Pa; Pa]}, \tag{1.1}$$

kde p_0 je referenční úroveň tlaku (práh slyšení): $2\cdot10^{-5}$ Pa. Pro hladinu intenzity zvuku se vztah upraví následovně (dle [6]):

$$L = 10\log(I/I_0)\ \text{[dB; W/m}^2\text{; W/m}^2\text{]}, \tag{1.2}$$

kde I_0 je referenční úroveň intenzity, I_o=$10^{-}12$ Wm^{-2}.

Vlastnosti lidského sluchu je také vhodné zohlednit při korekci hlasitosti a lineární regulace proto není vhodná. V případě analogových obvodů NF techniky se

zpravidla pro korektory užívají potenciometry s logaritmickou odporovou dráhou. V digitálních audio systémech lze ovládání hlasitosti realizovat v krocích např. po jednom decibelu.

1.2.4 Skládání tónů

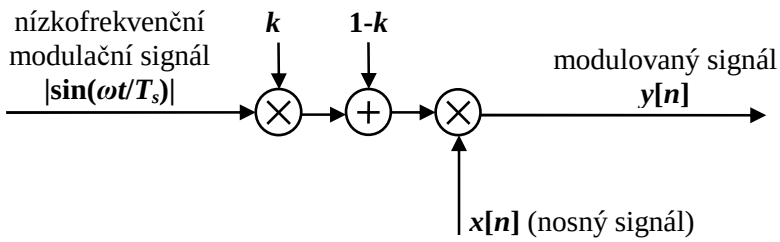
Výška tónu hraje rovněž důležitou roli při skládání více tónů. Současně hrající tóny, jejichž výšku lze vyjádřit poměrem malých celých čísel, znějí při poslechu zpravidla příjemně (jsou v harmonii). Naopak tóny, které nejsou v blízké příbuznosti, vyvolávají při poslechu nepříjemné pocity, říkáme, že jsou v disonanci. Souzvuk tří a více tónů se označuje jako akord [6].

2 Principy vybraných audio efektů

2.1 Tremolo

Efekt tremolo lze charakterizovat jako periodické kolísání hlasitosti zvukového signálu, jedná se tedy o amplitudovou modulaci (AM). Při běžném užití AM v radiotechnice je obvykle vysokofrekvenční nosný signál modulován užitečným signálem. Naproti tomu u efektu tremolo se jako nosný signál používá zvukový signál, který je následně modulován nízkofrekvenčním signálem o frekvenci v řádech jednotek Hz. Součtové a rozdílové frekvenční složky signálu vniklé modulací proto nejsou sluchem registrovány a výsledný efekt je vnímán jako periodické kolísání hlasitosti (viz [1]).

Tremolo se hodí zejména k modifikaci zvuku strunných nástrojů nebo lidského zpěvu a často se také kombinuje s vibrátem, které lze vytvořit pomocí fázové modulace, viz. kapitola 2.3.



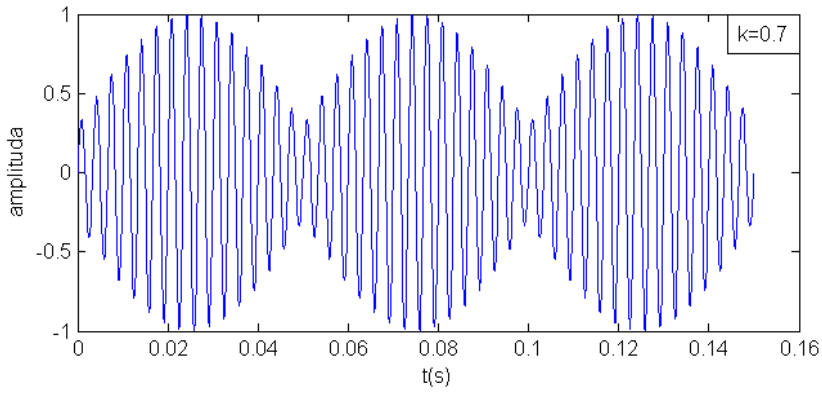
Obr. 2.1: principiální schéma efektu tremolo

Jednou z možných variant, jak realizovat efekt tremolo, popisuje rovnice 2.1.

$$y[n] = x[n] \cdot (1 - k + k \cdot \left|\sin(\pi n f_{MOD} \cdot nT_s)\right|) \tag{2.1}$$

Parametr k udává hloubku modulace, parametr f_{MOD} modulační frekvenci a T_S značí vzorkovací periodu.

Na obrázku 2.2 je příklad harmonického signálu o frekvenci 300 Hz modulovaného dle vztahu 2.1, modulační frekvencí 20 Hz s hloubkou modulace k = 0,7.



Obr. 2.2: Tremolo – amplitudová modulace signálu

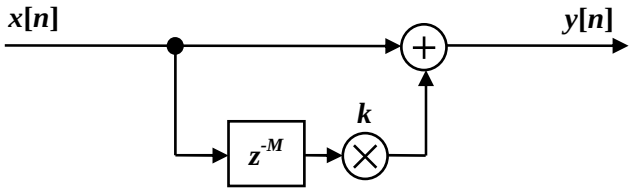
2.2 Delay, reverb

V české terminologii se oba efekty obecně označují jako ozvěna. Efekt delay lze jednoduše označit jako „zpoždění“ a efekt reverb jako „dozvuk“ (viz starší česká publikace [6]). Prvně zmiňovaný efekt vzniká např. při odrazu zvukové vlny od několik desítek až stovek metrů vzdálené překážky (např. skály). Ke zdroji se tedy vrátí s určitým zpožděním (stovky ms) a útlumem a posluchač zde vnímá ozvěnu. Naproti tomu dozvuk vniká např. v malé místnosti, kde se zvukové vlny odráží od blízkých stěn a překážek i několikanásobně ale po kratší dobu. Tyto efekty se používají zejména pro navození atmosféry většího poslechového prostoru a hodí se ke všem hudebním nástrojům, zpěvu nebo pro libovolné zvuky.

Základní strukturu delay efektu zobrazuje obrázek 2.3. Jedná se o filtr FIR s konečnou impulsní odezvou a přenosovou funkcí popsanou vztahem:

$$H(z) = 1 + k \cdot z^{-M},\ k \leq 1 \qquad (2.2)$$

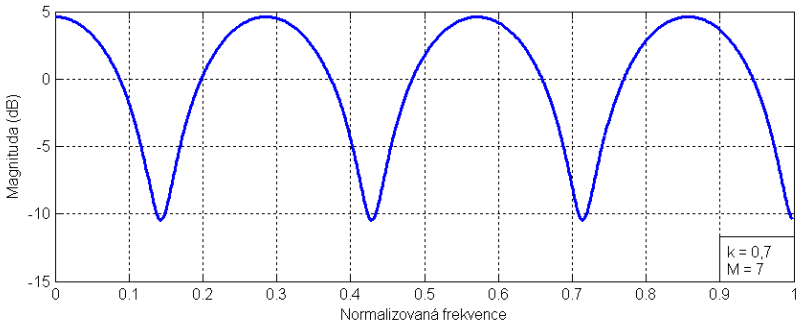
Hlasitost signálu zpožděného o *M* vzorků lze nastavit parametrem *k*.



Obr. 2.3: Základní struktura delay efektu – hřebenový FIR filtr

V zahraniční publikaci [1] je tento systém nazýván také jako hřebenový FIR filtr, jeho frekvenční amplitudová charakteristika totiž připomíná hřeben, což je znázorněno

na obrázku 2.4. Pro velké hodnoty *M*, které vytváří zpoždění přibližně větší než 50 ms, již nevnímáme změny ve frekvenční charakteristice ale časový posun (ozvěnu).

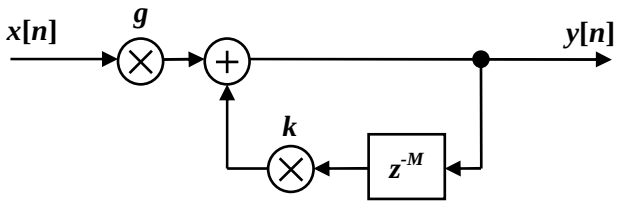


Obr. 2.4: Amplitudová charakteristika hřebenového FIR filtru

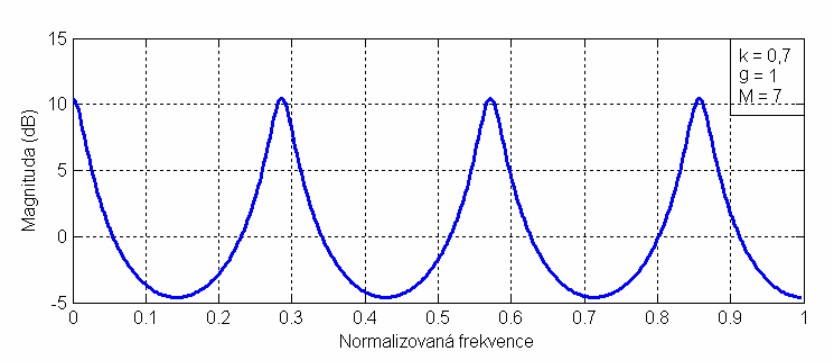
Pro realizaci vícenásobné ozvěny stačí do systému přidat další větve s různým zpožděním z^{-M}. Tím ale značně rostou i nároky na paměť použitého HW. Např. pro realizaci zpoždění 0,5 s při vzorkovací frekvenci 44,1 kHz je potřeba ukládat do paměti 22050 vzorků (M=22050). Výhodné je proto použít variantu efektu s hřebenovým filtrem IIR, který má nekonečnou impulsní odezvu díky zpětné vazbě. Filtr je znázorněn na obrázku 2.5. Přenosová funkce je definovaná následující rovnicí.

$$H(z) = \frac{g}{1 - k \cdot z^{-M}} \tag{2.3}$$

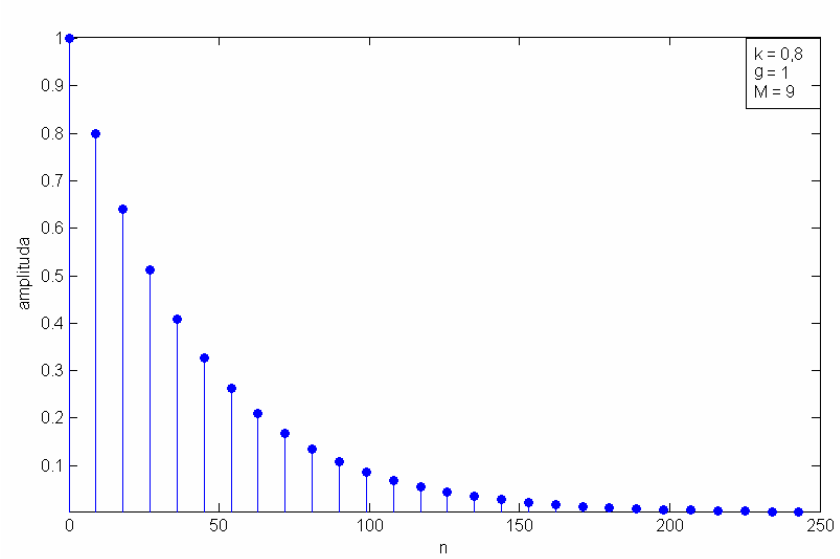
Parametr k zde oproti FIR variantě krom hlasitosti ozvěny ovlivňuje i délku jejího trvání. Pro zajištění podmínky stability systému dále musí platit: $|k| < 1$. Čím více se parametr k blíží jedné, tím delší je impulsní odezva systému a tedy i doznívání ozvěny, zároveň se tím zvyšuje celková hlasitost signálu. Pro korekci hlasitosti tak slouží parametr g. Frekvenční charakteristiku zobrazuje obr. 2.6, příklad impulsní odezvy s činitelem zpětné vazby k = 0,7 lze vidět na obr. 2.7. Pokud se podíváme na impulsní odezvu, je patrné, že amplituda zpožděných vzorků exponenciálně klesá.



Obr. 2.5: Struktura delay efektu s hřebenovým IIR filtrem



Obr. 2.6: Amplitudová charakteristika hřebenového IIR filtru



Obr. 2.7: Impulsní odezva hřebenového IIR filtru

Kombinací několika IIR hřebenových filtrů s různými koeficienty je možné zjednodušeně napodobit imitaci dozvuku v místnosti, další možnosti realizace lze nalézt v [1].

2.3 Efekty využívající variabilního zpoždění signálu

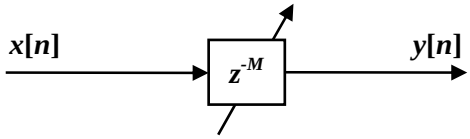
Pokud konstantní zpoždění u efektu delay z kapitoly 2.2 nahradíme zpožděním, které se mění v čase, získáme následující skupinu audio efektů. Jedná se o fázovou modulaci signálu, kde výsledný zvukový efekt závisí na průběhu modulačního signálu, zpětné vazbě, případně na dalších parametrech.

2.3.1 Vibráto

Princip samotného efektu *vibráto* zachycuje obrázek 2.8. Změna zpoždění (fáze) signálu vytváří efekt měnící se frekvence signálu (výšky tónu), stejně jako když např. kytarista napíná a povoluje struny na kytaře. Tento jev, podrobněji popsaný např. v [1], vzniká na podobném principu jako Dopplerův jev. Modulační signál má zpravidla harmonický průběh o frekvenci v jednotkách Hz. Přenosová funkce pro aktuální hodnotu zpoždění *M* vypadá následovně:

$$H(z) = z^{-M} \tag{2.4}$$

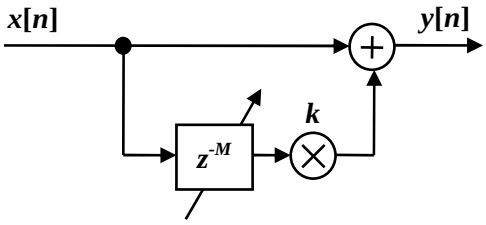
Vibráto se stejně jako tremolo používá zejména v kombinaci se strunnými nástroji nebo lidským zpěvem.



Obr. 2.8: Princip efektu vibráto

2.3.2 Flanger, chorus

Pokud fázově modulovaný signál sloučíme s původním signálem, získáme efekty flanger, chorus, případně jejich variace s vibrátem, viz obr. 2.9. Tyto efekty nejsou exaktně definovány a výsledný efekt závisí vždy zejména na nastavených parametrech a dané struktuře (více viz [1]). Pro aktuální hodnotu zpoždění *M* platí přenosová funkce hřebenového FIR filtru (vztah 2.2).



Obr. 2.9: Základní struktura efektů chorus, flanger, (vibráto)

Pro vytvoření tzv. čistého efektu *chorus* („*bílý/white chorus*“) lze dle [1] použít jako modulační signál nízkofrekvenční šum. Při modulaci harmonickým průběhem vzniká kombinace efektů chorus a vibráto. Efekt chorus se často používá např. ke „zjemnění“ zvuku akustické i elektrické kytary.

Naopak *flanger* se vytvoří pomocí modulace harmonickým signálem o velmi nízkém kmitočtu (0,1 až 1 Hz), tím je docíleno vjemu pomalu měnící se frekvence zvuku.

2.4 Efekty realizované pomocí filtrů

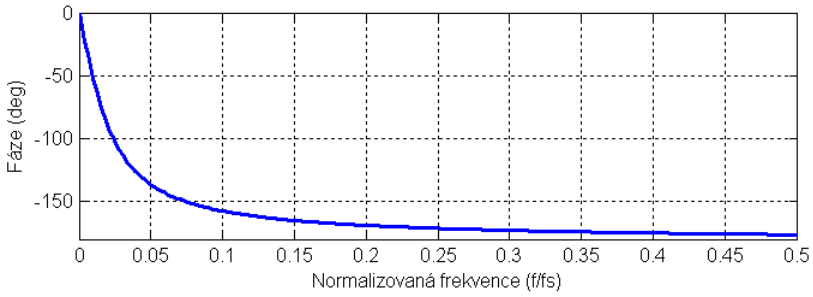
Filtry patří mezi nejčastěji používané systémy v audiotechnice. Obecné filtry se dělí podle průběhu amplitudové charakteristiky na dolní, horní propust (LP, HP), pásmovou propust (BP) a pásmovou zádrž (BR). V oblasti číslicového zpracování signálů rozlišujeme filtry s konečnou impulsní odezvou (FIR) a nekonečnou impulsní odezvou (IIR). Typ filtru vybíráme podle amplitudové charakteristiky, přičemž hlavní důraz je kladen na jednoduchost algoritmu. Naopak méně podstatný je průběh fázové charakteristiky. V oblastech, kde je požadována lineární fázová charakteristika, jsou s výhodou používány filtry FIR. Pro zpracování akustických signálů se však běžně používají filtry IIR, které mají při stejném řádu oproti filtrům FIR vyšší strmost. Nelineární průběh fáze nám nevadí, jelikož lidský sluch není schopen rozeznat tyto fázové rozdíly. Často se z historických důvodů také vychází z analogových filtrů, které lze v číslicové oblasti nahradit filtry IIR. Při implementaci si tak vystačíme s IIR filtry nízkých řádů (obvykle 1. až 6.), které jsou méně náročné na výpočetní výkon a kapacitu operační paměti použitého systému [1].

2.4.1 Allpass filtr

Mezi základní stavební prvky digitálních audio efektů patří krom obecně známých číslicových filtrů také tzv. *allpass filtr*. Allpass filtr prvního řádu je definován přenosovou funkcí [1]:

$$A(z) = \frac{c + z^{-1}}{1 + cz^{-1}} \tag{2.5}$$

Amplitudová charakteristika filtru je konstantní, fázová charakteristika se blíží k -180° pro vysoké frekvence, viz následující obrázek:



Obr. 2.10: Fázová charakteristika allpass filtru 1. řádu pro $f_c = 0{,}01fs$

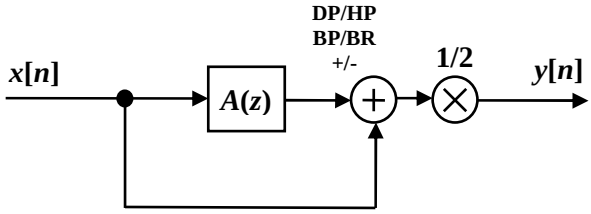
Této vlastnosti se využívá pro sestavení filtrů dolní a horní propust pomocí allpass filtru:

$$H(z) = \frac{1}{2}(1 \pm A_z), \qquad \text{(LP/HP +/-)} \tag{2.6}$$

Použití allpass filtru je výhodné zejména v případech, kdy chceme filtr přelaďovat v reálném čase. Pro nastavení mezní frekvence f_c stačí dopočítat jediný koeficient c, dle následujícího vztahu:

$$c = \frac{\tan(\pi f_C / f_S) - 1}{\tan(\pi f_C / f_S) + 1}, \tag{2.7}$$

kde f_s značí vzorkovací frekvenci. Přepočet mezní frekvence je tedy méně náročný než u běžných IIR filtrů, kde je potřeba přepočítávat několik koeficientů (v případě IIR filtru 1. řádu celkem čtyři koeficienty). Následující obrázek zobrazuje strukturu filtru realizovaného z allpass filtru:



Obr. 2.11: Realizace filtrů LP, HP, BP, BR pomocí allpass fitru

Pásmovou zádrž a pásmovou propust lze vytvořit stejným způsobem dle obr. 2.11 pomocí allpass filtru 2. řádu, který je popsán přenosovou funkcí:

$$A(z) = \frac{-c + d(1-c)z^{-1} + z^{-2}}{1 + d(1-c)z^{-1} - cz^{-2}} \tag{2.8}$$

Přenosovou funkci pásmové propusti a zádrže popisuje přenosová funkce:

$$H(z) = \frac{1}{2}(1 \pm A(z)), \quad \text{(BP/BR, +/-)}, \tag{2.9}$$

kde koeficient *c* ovlivňuje šířku frekvenčního pásma filtru f_B danou vztahem:

$$c = \frac{\tan(\pi f_B / f_S) - 1}{\tan(2\pi f_B / f_S) + 1} \tag{2.10}$$

Mezní frekvence f_C určuje koeficient *d*, který je definován vztahem:

$$d = -\cos(2\pi f_C / f_S) \tag{2.11}$$

Při ladění filtru je tedy potřeba přepočítávat pouze dva koeficienty.

2.4.2 Ekvalizér

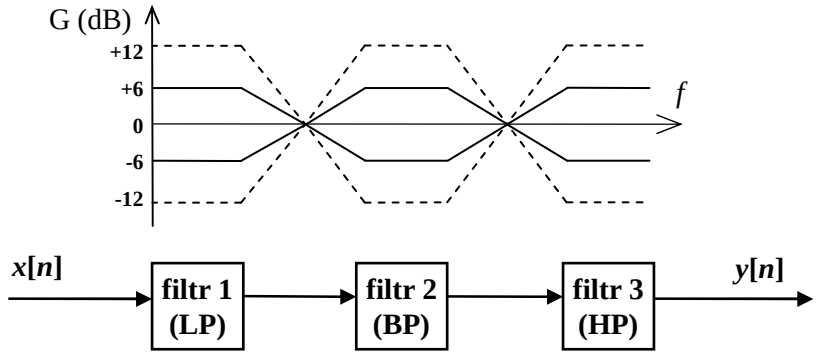
Princip ekvalizéru spočívá v rozdělení celého pásma akustických frekvencí na větší počet pásem, ve kterých je možné nezávisle na ostatních měnit úroveň signálu. Pomocí ekvalizéru lze např. nastavit optimální frekvenční charakteristiku nedokonalého elektroakustického řetězce nebo částečně kompenzovat nedokonalou akustiku ozvučovaného prostoru v závislosti na vkusu posluchače. V ozvučovací oblasti se ekvalizéry používají také např. k redukci akustické zpětné vazby nebo k úpravě barvy tónu hudebních nástrojů (viz kapitola 1.2.2).

Ekvalizér lze sestavit pomocí filtrů definovaných přenosovou funkcí:

$$H_{EF}(z) = 1 + g \cdot H(z), \tag{2.12}$$

kde *H(z)* je přenosová funkce zvoleného obecného filtru (LP, BP nebo HP), který může být realizován pomocí výše uvedených allpass filtrů, případně jiných IIR či FIR filtrů popsaných např. zde: [3]. Parametr *g* určuje zesílení daného frekvenčního pásma. Pro záporné hodnoty *g* dochází k potlačení stejných frekvencí.

Na obrázku 2.12 je naznačena realizace ekvalizéru pomocí sériově řazených filtrů. Naznačena je rovněž možnost korekce frekvenční amplitudové charakteristiky. Obvykle se jednotlivá frekvenční pásma překrývají, zde jsou však pro přehlednost znázorněny odděleně. Rovněž lze použít i jiné struktury s paralelním či sério–paralelním řazením filtrů, jejichž realizaci lze nalézt například v [1].

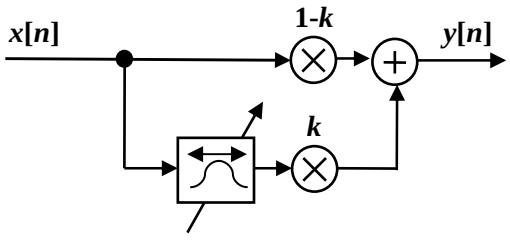


Obr. 2.12: Sériová struktura ekvalizéru

2.5 Efekty realizované pomocí laditelných filtrů

2.5.1 Wah-wah

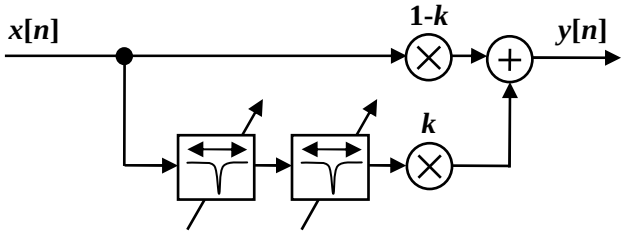
Wah-wah („*kvákadlo*“) patří do skupiny efektů využívajících spektrální modulace signálu [6]. Lze jej vytvořit pomocí filtru typu pásmová propust, jehož mezní frekvence se přelaďuje v reálném čase, ve frekvenčním rozsahu přibližně od 200 Hz do 2 kHz. Princip je patrný z obr. 2.13. Mezní frekvence filtru se většinou mění pomocí nohou ovládaného pedálu, nebo též v závislosti na intenzitě signálu, případně se moduluje periodickým harmonickým průběhem. Získaný efekt, který mění signál automaticky, se často nazývá jako *auto-wah*. Absolutní šířka pásma filtru se obvykle mění zároveň s mezní frekvencí, tak aby zůstal poměr šířky/frekvence přibližně stejný. Upravený signál se slučuje s původním signálem a podle poměru těchto dvou složek se nastavuje intenzita výsledného efektu, což lze provést změnou parametru *k* v rozsahu hodnot 0 až 1.



Obr. 2.13: Princip efektu wah-wah

2.5.2 Phaser

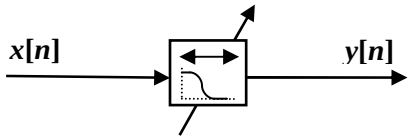
Efekt *phaser* získáme stejným způsobem jako wah-wah, pokud zaměníme filtr typu pásmová propust za úzkou pásmovou zádrž. Pro bohatší efekt lze zařadit do série několik pásmových zádrží s různou mezní frekvencí, jak je znázorněno na obrázku 2.14.



Obr. 2.14: Princip efektu phaser

2.5.3 Variabilní dolní propust

V některých případech lze využít i přeladitelný filtr typu dolní propust. Pokud se mezní frekvence filtru mění periodicky, získáme efekt používaný např. v elektronických hudebních stylech. Modulační frekvence se volí nízká, obvykle v rozsahu 0,1 Hz až 1 Hz. Charakter efektu lze přirovnat např. k poslechu zdroje zvuku, který je umístěn za postupně se odsouvající překážkou. Překážka mezi zdrojem a posluchačem nemá velký vliv na nízké kmitočty, ale působí vysoký útlum na vyšší kmitočty, které se šíří více směrově [7].



Obr. 2.15: Variabilní dolní propust

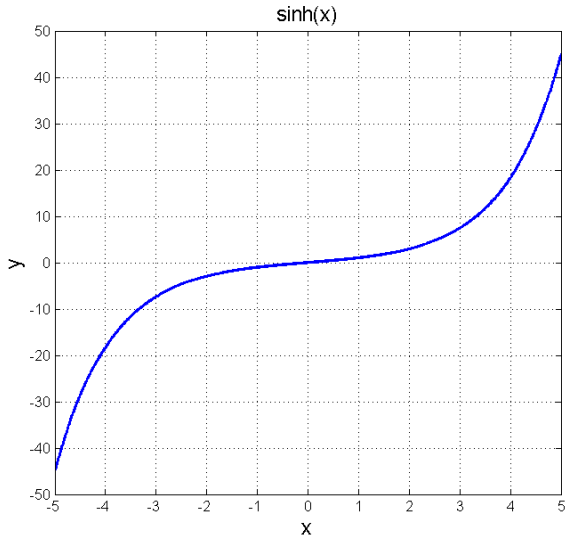
2.6 Nelineární audio efekty

Mezi nelineární efekty zahrnujeme zejména algoritmy pro ovlivnění dynamiky audio signálu, efekty imitující přebuzení zesilovače nebo např. simulátory elektronkových zesilovačů. Avšak některé další efekty, které jsou z principu také nelineární (neplatí princip superpozice), nezahrnujeme do kategorie nelineárních efektů. Příkladem může být např. již popsaná varianta efektu wah-wah (kapitola 2.5.1), jehož spektrum je modulováno laditelným filtrem v závislosti na intenzitě signálu.

V oblasti číslicových signálů lze podobné algoritmy popsat následujícím vztahem,

$$y = f(x[n]), \tag{2.13}$$

kde $f(x[n])$ je libovolnou nelineární funkcí vzorkované proměnné $x[n]$. Např. při realizaci kompresoru dynamiky lze použít logaritmickou funkci. Jiným příkladem nelineární funkce může být hyperbolický sinus, jehož průběh je zobrazen na obr. 2.16. Podobná funkce se hodí k realizaci dynamického expandéru.



Obr. 2.16: Příklad průběhu nelineární funkce sinus hyperbolický

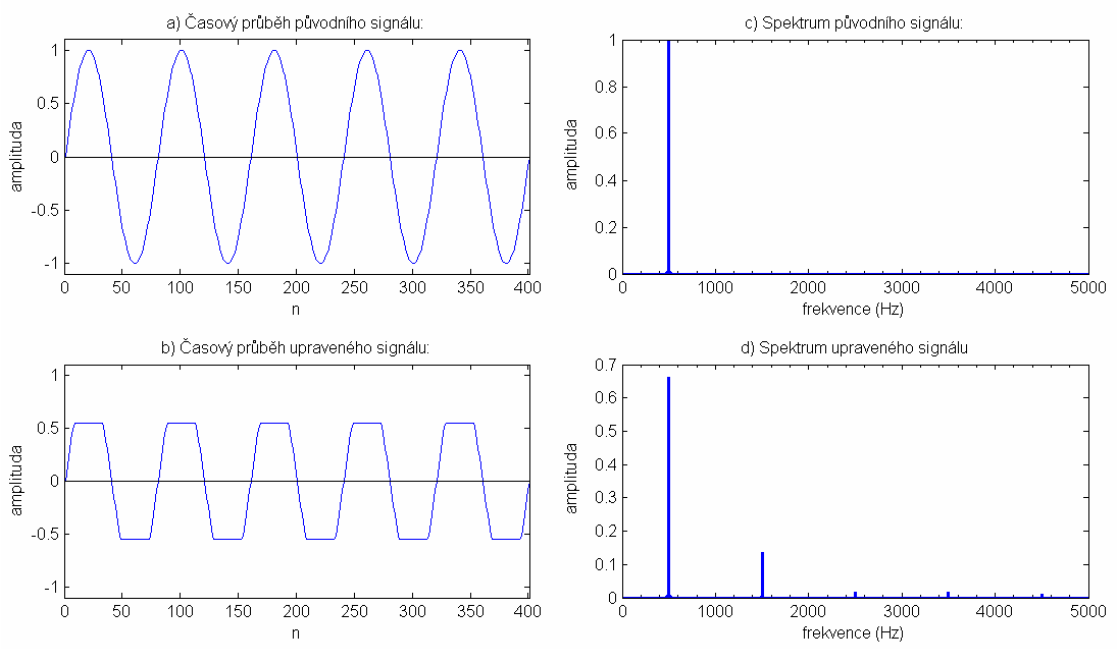
Dle teorie Fourierovy analýzy lze libovolný signál rozložit na řadu harmonických signálů. Lineární časově invariantní systémy (LTI), mezi něž patří i filtry, mohou ovlivnit pouze amplitudu případně fázi jednotlivých harmonických složek. Naproti tomu nelineární systémy jsou schopny generovat i nové harmonické složky, jak je popsáno níže [1].

2.6.1 Zkreslení

Zkreslený zvuk elektrické kytary tvoří základ rockové hudby. Zejména z historického hlediska se rozlišuje několik základních typů zkreslení. Pokud lehce přebudíme elektronkový či tranzistorový zesilovač, dojde k limitaci výstupního signálu, jelikož výstupní rozkmit signálu je omezen napájecím napětím zesilovače. Takto získané zkreslení se označuje jako *overdrive*. Efekt *distortion* je naproti tomu výraznější. V analogových obvodech se vytváří silným zesílením kytarového signálu (*boost*) a následnou limitací pomocí obvodu vytvořeného např. pomocí antiparalelně zapojených diod (*clipping*), čímž se docílí výrazného zkreslení, a výstupní signál obsahuje velký počet vyšších harmonických složek. Trochu odlišného efektu se

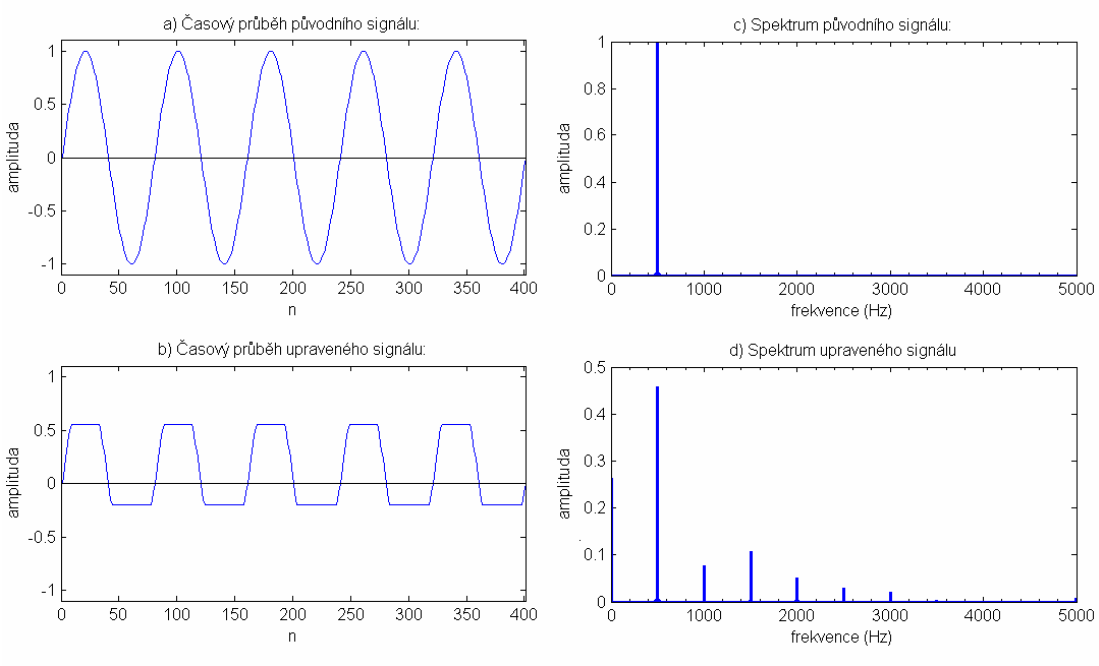
dosáhne, pokud jsou prahy pro limitaci kladných a záporných hodnot různé, podobný efekt se označuje jako *fuzz*.

Na obrázku 2.17 je uveden příklad zkreslení vytvořeného pomocí *symetrické limitace* signálu. Nahoře je znázorněn původní harmonický signál o jednotkové amplitudě, frekvenci 500 Hz a dále jeho frekvenční spektrum. Pod ním je zobrazen signál, jehož amplituda je limitována na hodnotou 0,55 (v obou příkladech byla použita vzorkovací frekvence 44 kHz). Ve spektru upraveného signálu lze spatřit nové, vyšší harmonické složky, které jsou násobky základní harmonické. V tomto případě se jedná o liché násobky základní frekvence ($3 \cdot 500, 5 \cdot 500, \ldots$ Hz).



Obr. 2.17: Symetrická limitace harmonického signálu (*distortion*)

Jak bylo naznačeno výše, odlišného efektu se dosáhne v případě, kdy je signál limitován asymetricky, což lze vidět na obrázku 2.18. Frekvenční spektrum zkresleného signálu obsahuje sudé i liché násobky první harmonické frekvence a rovněž nežádoucí stejnosměrnou složku, kterou je možné odfiltrovat strmým HP filtrem.



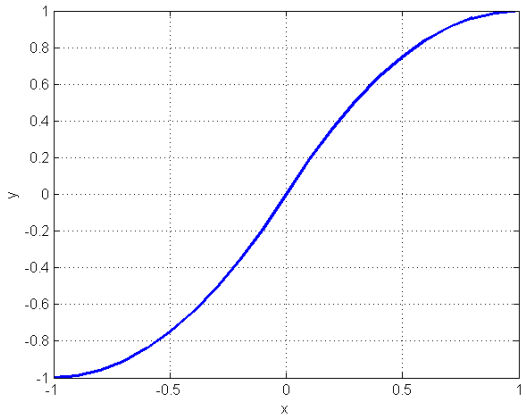
Obr. 2.18: Asymetrická limitace harmonického signálu (*fuzz*)

2.6.2 Simulace elektronkového zesilovače

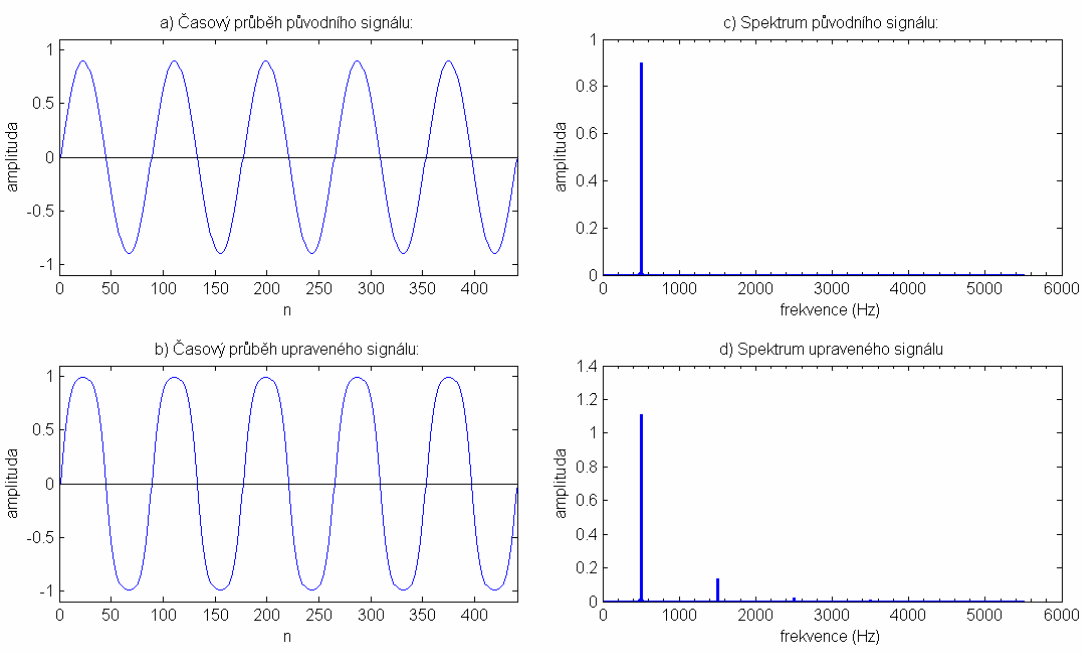
Vhodnou nelineární funkcí lze přibližně simulovat i elektronkové zesilovače. Na obrázku 2.19 je zobrazena křivka kvadratické funkce definovaná na dvou intervalech:

$$x > 0:\ y = ax - bx^2,\ x < 0:\ y = ax + bx^2,\ a = 2, b = 1 \qquad (2.14)$$

Křivka přibližně aproximuje volt-ampérovou (VA) charakteristiku elektronkového zesilovacího stupně. V publikaci [1] lze nalézt i další možné funkce, jimiž lze simulovat různé elektronkové obvody. Na obr. 2.20 můžeme vidět, jaký vliv má nelineární funkce na frekvenční spektrum. Je patrné, že zkreslení signálu je menší a celkový podíl nových harmonických složek je nižší než-li v případě ostré limitace zobrazené na obr. 2.17 a 2.18.



Obr. 2.19: Aproximace VA char. elektronkového zesilovače kvadratickou funkcí



Obr. 2.20: Simulace elektronkového zesilovače

Podrobnou simulací elektronkového zesilovače na základě diferenciálního popisu se zabývá článek [14].

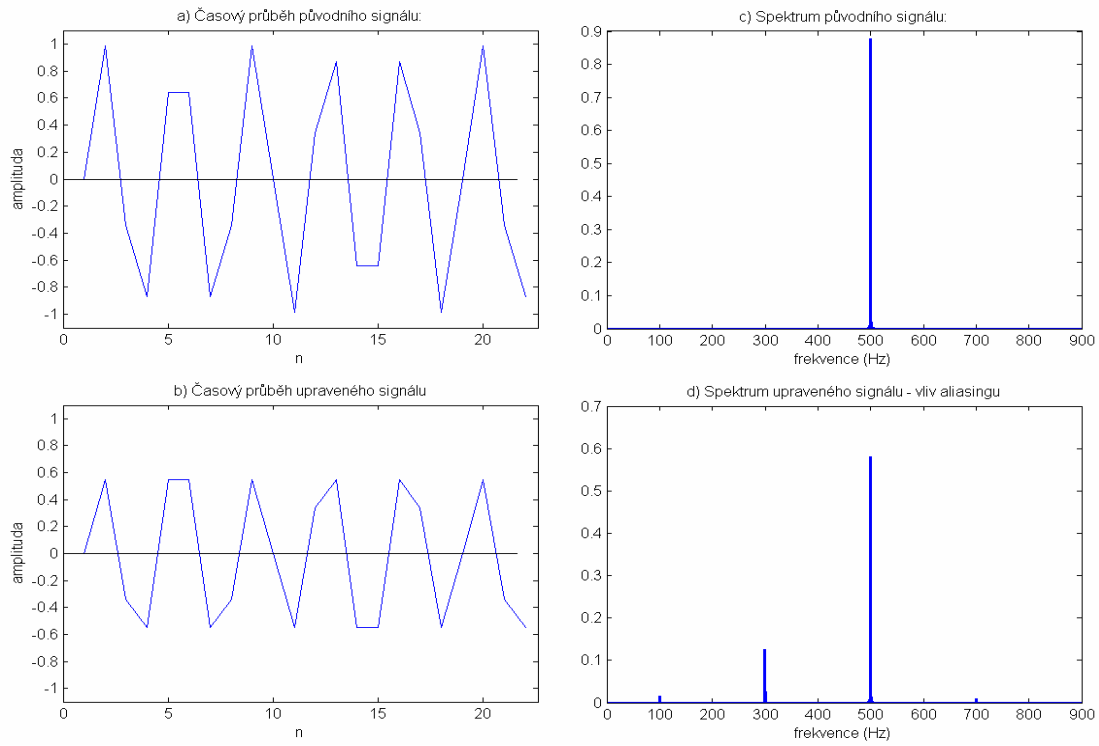
2.6.3 Vliv aliasingu na efekty zkreslení

Při zavedení zkreslení v digitálních systémech můžeme narazit na problém způsobený aliasingem. Pro nově vytvořené vyšší harmonické složky již totiž nemusí platit *Shannonův vzorkovací teorém* ($f < 2f_S$). V takovém případě dojde k překladu vyšší harmonické složky o frekvenci f na jinou frekvenci f_a, která leží v pásmu $0 < f_a < f_S/2$. Nepříjemné je, že takto nově vzniklé frekvenční složky ve většině případů nejsou celočíselným násobkem původní frekvence a výsledný signál proto není v harmonii a může působit zvukově nepříjemně (viz kapitola 1.2.4).

Vznik zkreslení způsobené aliasingem ilustruje obrázek 2.21. Jedná se o symetrickou limitací harmonického signálu o frekvenci 500 Hz, stejným způsobem jako na obrázku 2.17, avšak záměrně zde byla použita nízká vzorkovací frekvence 1900 Hz. Ve spektru vidíme, že původní harmonická složka o frekvenci 500 Hz ještě splňuje vzorkovací teorém, ale nově vytvořené složky již byly přeloženy. Konkrétně třetí harmonická, který by měla mít frekvenci 1500 Hz, byla posunuta na 300 Hz.

Při realizaci zkreslení v číslicových systémech je tedy vhodné pro omezení aliasingu používat výrazně vyšší vzorkovací frekvence, než definuje vzorkovací teorém. Pokud to daný HW neumožňuje, nabízí se možnost omezit spektrum původního signálu

LP filtrem, případně nahradit ostrou limitaci plynulejší funkcí (jako na obr. 2.19). Tím však změníme i charakter zvuku výsledného efektu (viz kapitola 4.4.1). Jiným řešením může být využití techniky, která zahrnuje převzorkování signálu na vyšší frekvenci (upsampling), následnou aplikaci nelineárních efektů a poté zpětné podvzorkování na původní vzorkovací frekvenci (downsampling) [1].

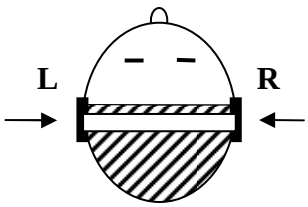


Obr. 2.21: Vliv aliasingu na efekt zkreslení při nízké vzorkovací frekvenci (1,9 kHz)

2.7 Stereofonní efekty

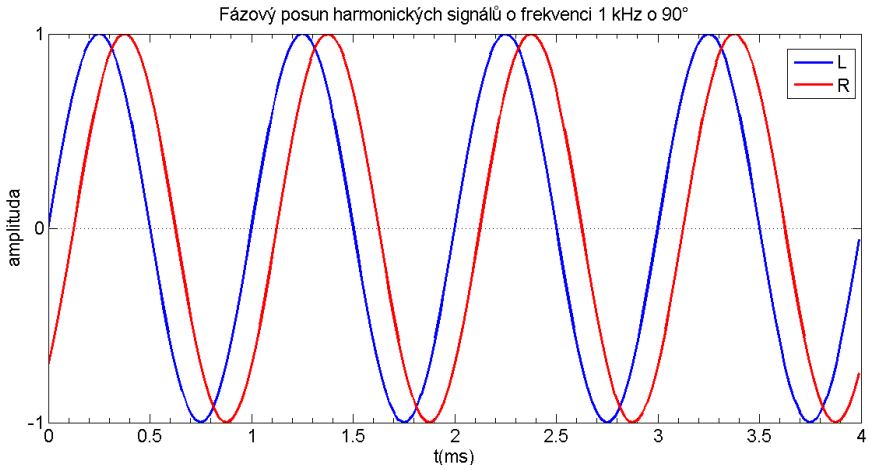
Stereofonní poslech hudebních signálů je oproti monofonnímu přirozenější a líbivější. Dvoukanálový záznam i reprodukce hudby se proto staly standardem již v druhé polovině 20. století. Ačkoliv pro ozvučení filmů se postupně přešlo k vícekanálové prostorové projekci, při poslechu hudby se stále upřednostňuje kvalitní dvoukanálová aparatura nad poslechem základních vícekanálových systémů. Naopak jednokanálové zpracování akustických signálů se stále používá zejména při přenosu a zpracování lidské řeči, kde nejsou tak vysoké nároky na kvalitu reprodukce a omezen je rovněž frekvenční rozsah. Např. v oblasti telekomunikací se pracuje s frekvenčním pásmem od 300 do 3400 Hz.

Při realizaci základních stereofonních efektů se vychází zejména z vlastností lidského sluchu a principu prostorového slyšení. Klíčovou vlastností je v tomto ohledu zejména velmi dobrá schopnost vnímání časových a fázových diferencí mezi levým a pravým sluchovým orgánem. Zvuk, který k posluchači přichází např. zleva, dorazí k levému uchu dříve než k pravému. Pro frekvenci 1000 Hz lze zpozorovat časovou diferenci 10 až 15 μs [7], tato citlivost dále závisí na délce trvání vyhodnocovaného zvuku. Při délce trvání kratší než cca 250 ms se práh rozpoznatelné časové odchylky zvyšuje. Při prostorovém slyšení se dále velkým podílem uplatňují také rozdíly v hlasitosti a frekvenci mezi levým a pravým sluchovým orgánem. Pokud přichází zvuková vlna k posluchači z boku, na jedno ucho dopadá přímo, kdežto na druhé s určitým útlumem, případně s upraveným frekvenčním pásmem. Hlava zde tvoří překážku, která působí na zvukovou vlnu jako filtr (dolní propust). Nízké kmitočty jsou potlačeny méně, jelikož se šíří všesměrově. Větší útlum je patrný u vyšších kmitočtů, které jsou směrovější [7].



Obr. 2.22: Stereofonní poslech pomocí sluchátek

Základní stereofonní efekty se tedy vytváří zavedením fázových diferencí mezi levým a pravým kanálem, případně doplněné frekvenčními filtry. Pro změnu lokalizace přehrávaného zvuku lze jednoduše použít změnu vyvážení (balance) mezi oběma kanály. Obrázek 2.23 ilustruje fázový posun dvou harmonických signálů o úhel 90°. Při frekvenci 1 kHz odpovídá rozdíl časovému posunu mezi levým a pravým kanálem o 250 μs. Tato diference vyvolá při poslechu silný stereofonní vjem.



Obr. 2.23: Časová diference mezi levým a pravým kanálem signálu

3 Použité HW a SW vývojové prostředky

Pro ověření principů nastudovaných efektů a rovněž návrh vlastních efektů bylo použito programové prostředí MATLAB, které je spolu s vlastním skriptovacím jazykem mimo jiné vhodné i pro zpracování signálů. MATLAB je velice výkonným nástrojem, který umožňuje poměrně snadnou a rychlou implementaci požadovaných algoritmů pro zpracování audio nahrávek uložených v paměti počítače. Cílem práce však byla zejména implementace efektů pracujících v reálném čase. Zapůjčena byla na fakultě dostupná vývojová deska (Starter kit) TMS320C6416 DSK (dále jen C6416 DSK) z produkce společnosti Spectrum Digital osazená signálovým procesorem Texas Instruments TMS320C6416T.

Zmíněný DSP pracuje pouze s pevnou desetinnou čárkou, avšak pro realizaci kvalitních digitálních audio efektů je vhodnější používat datové typy s plovoucí desetinnou čárkou (např. 32b float), které nabízejí větší dynamický rozsah a menší zaokrouhlovací chyby. Prostředí CCS je schopné kompilovat program z jazyka C používajícího datové typy s pohyblivou desetinnou čárkou i pro procesory řady C64xx, avšak za cenu vyšších časových nároků pro vykonání základních operací. Výkon použitého procesoru se však díky vysokému pracovnímu taktu (1 GHz) ukázal pro danou úlohu jako dostatečný, nicméně vhodnější by byl spíše procesor s HW podporou plovoucí desetinné čárky (např. TMS320C6713), zejména s ohledem na nižší spotřebu a případnou jednodušší optimalizaci algoritmů v assembleru, viz kapitola 3.3.2.

3.1 Digitální signálový procesor TMS320C6416

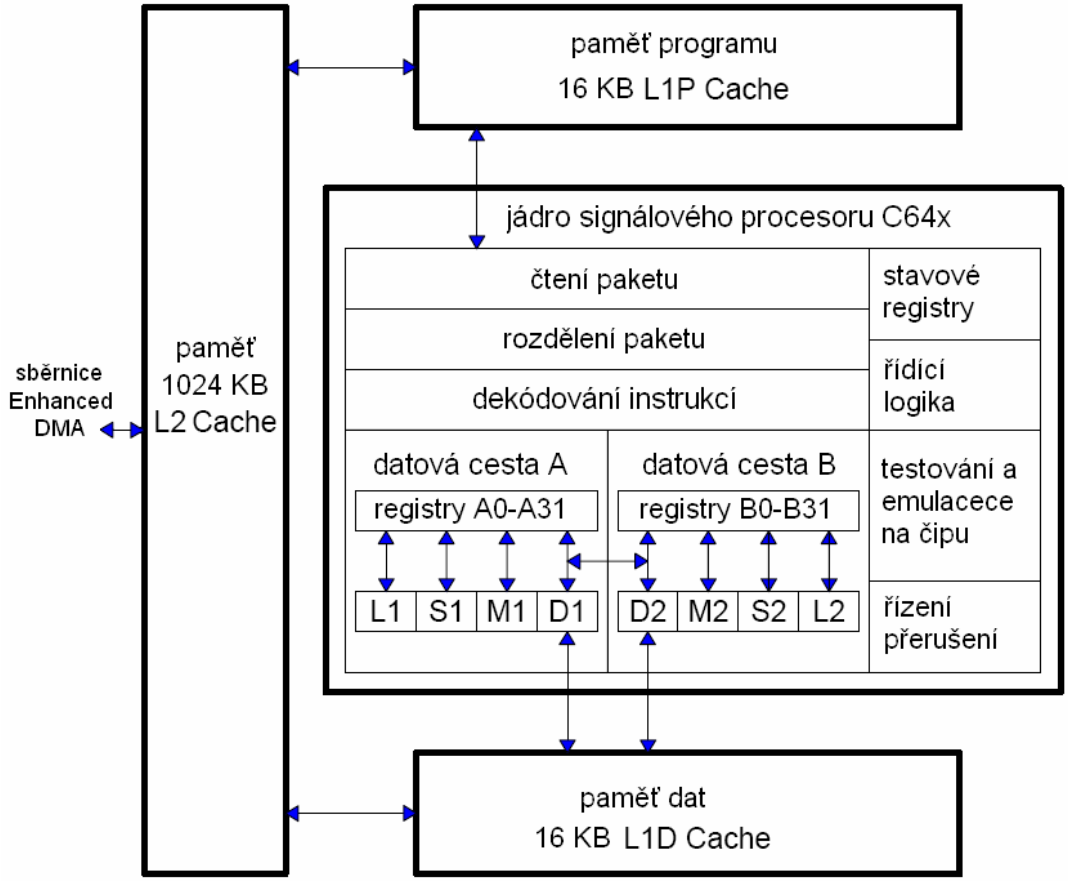
Pod pojmem digitální signálový procesor (DSP) rozumíme procesor, který je navržen pro rychlé a efektivní zpracování číslicových signálů, nejčastěji v reálném čase. Mezi typické úlohy zpracovávané pomocí DSP patří např. číslicové filtry, porovnávání signálů, kódování a dekódování nebo výpočet rychlé Fourierovy transformace (FFT). Architektury DSP jsou navrženy tak, aby podávaly vysoký výkon v podobných úlohách při zachování nízké ceny a spotřeby procesoru. Výrobci se proto snaží snižovat velikost čipu omezením velikosti paměti pro data i program. O to větší nároky jsou kladeny na kompilační programy, které musí generovat úsporný a efektivní strojový kód. Pro dosažení vyšších výkonů se DSP taktují vysokými kmitočty. Důležitou roli hraje rovněž velká míra paralelismu, případně zřetězené zpracování dat (tzv. pipelining). Instrukční

sada je optimalizovaná tak, aby časté operace jako součin s následnou akumulací mohly být provedeny v jednom instrukčním cyklu. Signálové procesory se v některých ohledech výrazně odlišují od procesorů pro všeobecné použití. V případě signálových procesorů architektury VLIW (viz níže) se rozhoduje o tom, které instrukce budou vykonány paralelně nebo zřetězeně, již v době kompilace. Díky tomu lze při ladění algoritmů určit časovou náročnost. Naproti tomu běžné procesory, které známe např. u osobních počítačů typu PC, jsou vybaveny HW jednotkami, které rozhodují o zřetězeném zpracování až za chodu programu, a stejný algoritmus tak může trvat pokaždé odlišnou dobu [8].

DSP firmy Texas Instruments řady TMS320C64xx (zkráceně C64x) s pevnou desetinnou čárkou využívají architektury typu VLIW (Very Long Instruction Word), označovanou jako VelociTI.2™. Procesory architektury VLIW se vyznačují tím, že obsahují několik funkčních jednotek, které jsou schopny nezávisle na sobě provádět různé operace. Tyto jednotky se řídí pomocí sdružených instrukcí, tzv. instrukčních paketů. Procesor je díky tomu schopen načíst i několik instrukcí najednou a následně je vykonat paralelně.

Na obr. 3.1 je zobrazeno blokové schéma jádra DSP C64x. Jádro obsahuje dva páry funkčních jednotek L, S, M a D, které vytváří dvě nezávislé datové cesty, A a B. Tato koncepce odpovídá obecné charakteristice paralelního systému typu SIMD (Single Instruction, Multiple Data). Jednotka L slouží především pro výpočet aritmetických operací, jednotka S pro bitové operace a větvení programu, jednotka M pro násobení a jednotka D umožňuje adresování a přístup do paměti. Některé instrukce však mohou být vykonány nezávisle na typu funkční jednotky, což je vylepšení oproti starší generaci signálových procesorů TMS320C62xx, které zajišťuje vyšší flexibilitu. Instrukční paket se může u procesorů řady C64x skládat až z osmi dílčích instrukcí, které jsou vykonány současně pomocí jednotlivých funkčních jednotek procesoru. Procesor je dále vybaven 64 32b registry všeobecného použití. Vyrovnávací paměť typu Cache je dvouúrovňová. Na čipu je integrováno 16 kB programové a 16 kB datové paměti L1 Cache. Paměť druhé úrovně o kapacitě 1024 kB je sdílená pro data i program a lze ji nakonfigurovat jako paměť typu RAM nebo L2 Cache. Šířka datové a adresové sběrnice činí 32 bitů, externí paměť RAM lze k procesoru připojit pomocí 64b sběrnice EMIF (External Memory Interface). Pro paměť programu případně konfigurační obvody, které nevyžaduje takovou přenosovou rychlost, lze použít 16b sběrnici EMIF. Maximální kapacita adresovatelného prostoru externí paměti činí 1024 MB. Mezi další periferie

patří např. řadič přímého přístupu do paměti (DMA) s propustností až 2 GB/s, sériová rozhraní, trojice časovačů, integrovaný oscilátor a rozhraní JTAG umožňující ladění a emulaci na čipu za chodu programu [8], [10].



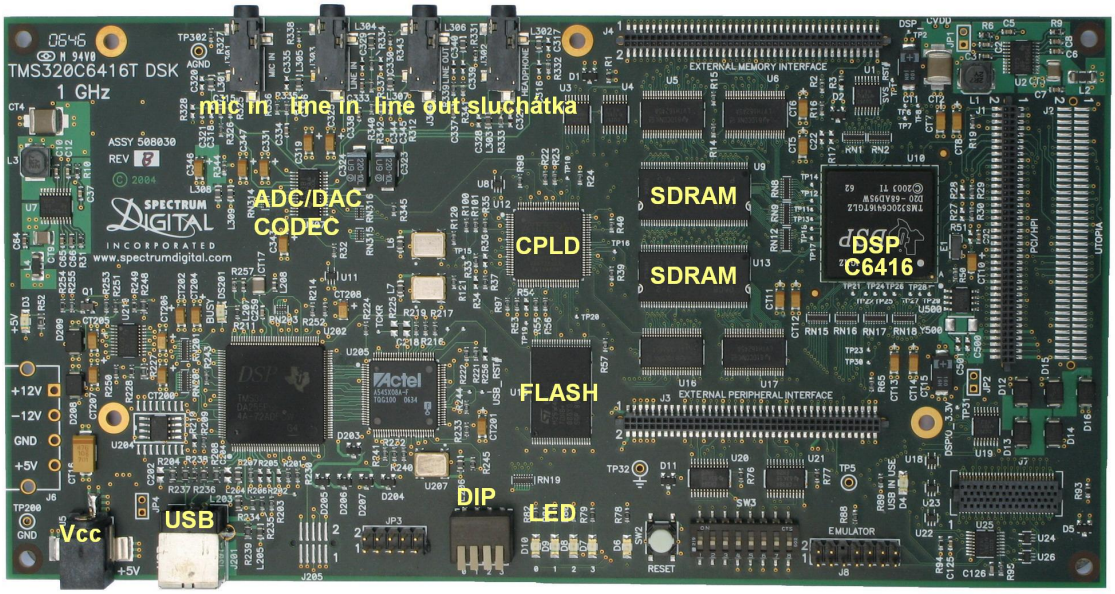
Obr. 3.1: Blokové schéma jádra procesoru TMS320C64xx

Obvod TMS320C6416T, jenž je součástí použité vývojové desky popsané v následující kapitole, je vyráběn technologií 0,09 μm a podporuje taktovací frekvence 600, 720, 850 až 1000 MHz. Délka jednoho instrukčního cyklu při taktu 1 GHz činí 1 ns, a procesor tak dosahuje teoretického výkonu až 8000 milionů instrukcí za sekundu (MIPS) [11]. Napájecí napětí vstupních a výstupních obvodů je 3,3 V. Samotné jádro je při taktu 1 GHz napájeno napětím 1,2 V a průměrná spotřeba při modelovém vytížení jádra (na 60 %) a periferií se pohybuje pouze kolem hodnoty 1,65 W [12]. Samotný čip tedy není potřeba chladit přídavným chladičem, jak dokládá fotografie vývojové desky v následující kapitole (obr. 3.2).

3.2 Vývojový deska C6416 DSK

Základem použitého vývojového prostředku C6416 (DSP Starter Kit) od společnosti Spectrum Digital je výše popsaný signálový procesor TMS320C6414T, taktovaný na frekvenci 1 GHz. Obr. 3.2 zobrazuje samotnou desku, kde jsou kromě signálového procesoru vyznačeny některé další obvody a I/O:

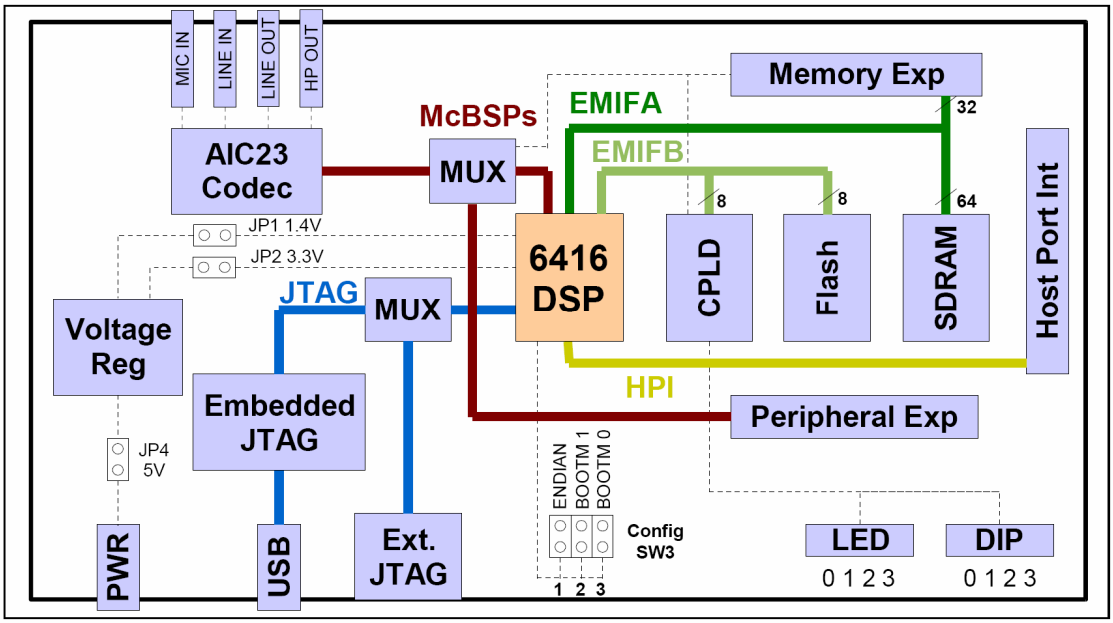
- AIC23 CODEC (AD a DA převodníky)
- 16 MB operační paměti SDRAM
- 512 kB programové paměti FLASH
- 4 programovatelné DIP spínače a 4 LED
- SW konfigurace pomocí registrů realizovaných obvodem CPLD
- JTAG emulace přes USB



Obr. 3.2: Vývojová deska C6416 DSK

Deska je dodávána zároveň se síťovým napájecím zdrojem 5 V a CD s vývojovým prostředím Code Composer Studio 3.1 včetně dalších knihoven a konfiguračních souborů. Pro připojení desky k PC slouží USB kabel.

Blokové schéma desky zobrazuje obr. 3.3. Kromě již popsaných obvodů je zde znázorněna také např. 64b sběrnice EMIFA, která slouží pro připojení externí operační paměti a 16b sběrnice EMIFB, která realizuje spojení s pamětí typu Flash a programovatelným obvodem CPLD. Zkratka McBSPs označuje vícekanálové sériové porty (Multichannel Buffered Serial Port).



Obr. 3.3: Blokové schéma vývojové desky C6416 DSK [5]

3.2.1 AD a DA převodníky

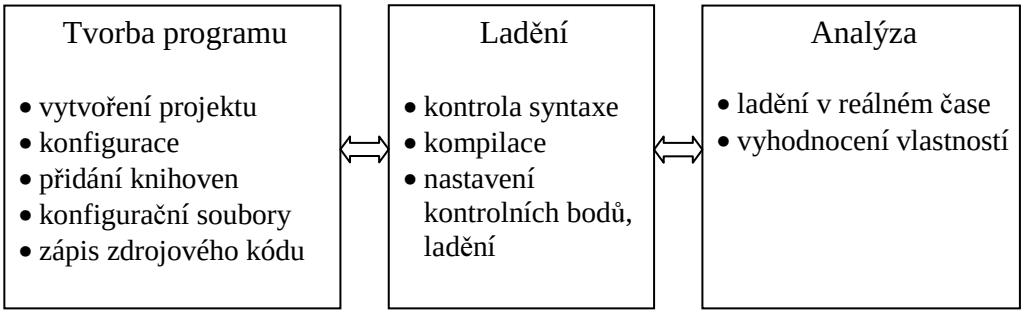
Použitá deska obsahuje stereo kodek *AIC23*, který realizuje AD i DA převodníky technologie *sigma–delta*. Součástí kodeku je i aliasingový filtr a rovněž filtr, který slouží pro rekonstrukci signálu při DA převodu. Kodek pracuje se vzorkovací frekvencí od 8 kHz do 96 kHz, kterou je možné softwarově konfigurovat v krocích (dostupné jsou např. běžně používané hodnoty 22 kHz, 44,1 kHz a 48 kHz). Šířka datového slova činí 16 bitů. Deska obsahuje dva analogové vstupy, mikrofonní (mono) a dále linkový stereoofonní vstup (line in), na který lze připojit běžné audio zařízení. Oba analogové výstupy jsou stereofonní. Opět zde nalezneme linkový výstup (line out) a dále sluchátkový výstup s nižší výstupní impedancí. Zmíněné analogové I/O jsou osazeny konektory Jack o průměru 3,5 mm.

3.3 Code Composer Studio

Code Composer Studio (CCS) je integrované vývojové prostředí pro DSP, mikrokontroléry a další aplikační procesory společnosti Texas Instruments. CCS zahrnuje nástroje pro programování, kompilaci a ladění programů a to jak na reálných procesorech tak pomocí simulátorů. V době psaní tohoto textu byla k dispozici verze CCS 4.0, která je založena na moderní vývojové platformě Eclipse. Toto open source prostředí je známé zejména programátorům v jazyce Java, avšak pomocí doplňků

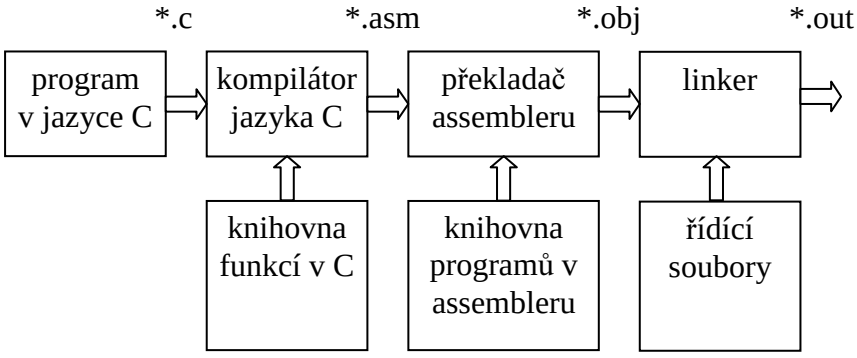
(pluginů) jej lze rozšířit i pro práci s dalšími programovacími jazyky. K použité vývojové desce C6416 DSK je přibalena starší verze programu CCS 3.1, avšak výhodou je, že součástí CD je i konfigurační instalátor, který nastaví CCS 3.1 pro práci se zmíněnou vývojovou deskou, aniž by bylo potřeba zdlouhavě konfigurovat celé prostředí. K dispozici je rovněž několik vzorových projektů, knihovny a konfigurační soubory pro inicializaci kodeku AIC23 a dalších periferií. Pro potřeby této práce bylo tedy použito CCS verze 3.1.

Tvorba programu v prostředí CCS sestává z několika kroků, které zachycuje následující schéma:



Obr. 3.4. Popis tvorby programu v prostředí Code Composer Studio

Pro programování je přítomen textový editor. Zdrojový kód je možné psát v programovacím jazyce assembler příslušného procesoru, standardem je však použití vyššího jazyka, C nebo C++. Obr. 3.5 zachycuje tvorbu programu pro signálový procesor:



Obr. 3.5. Tvorba programu pro DSP

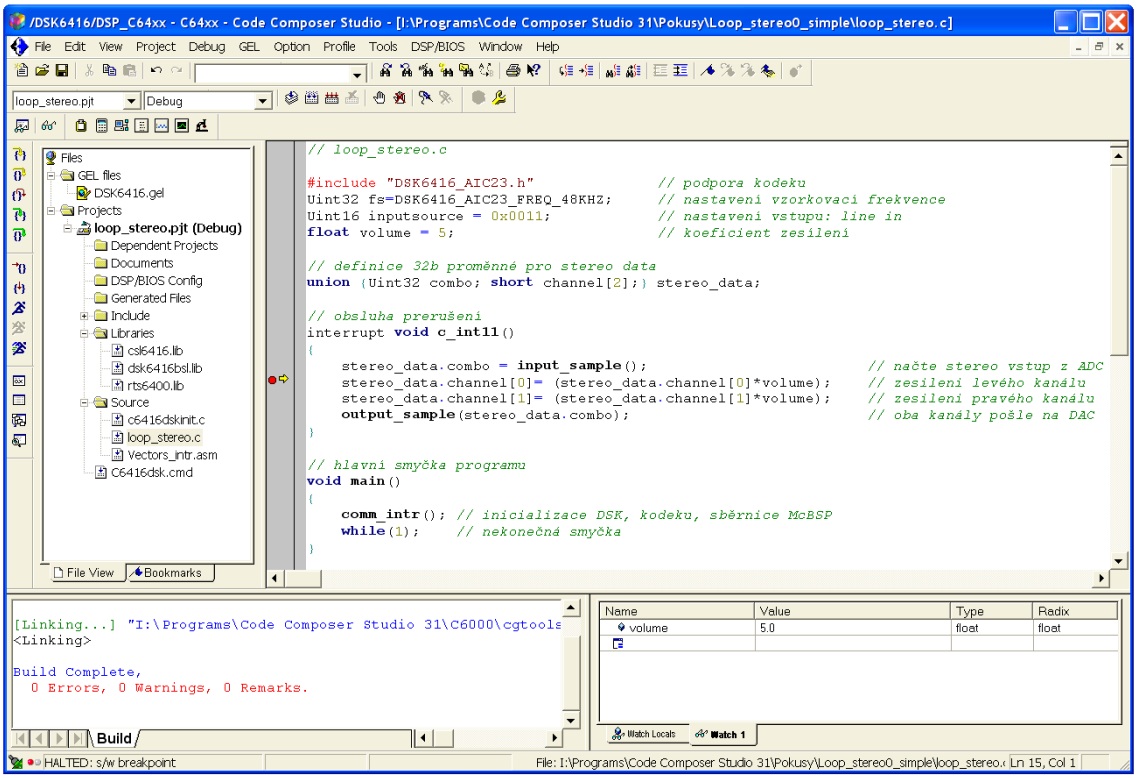
Zdrojové kódy v C jsou s využitím knihoven funkcí přeloženy kompilátorem do assembleru. Před kompilací je možné zvolit čtyři stupně optimalizace kódu. Při nastavení maximálního stupně optimalizace se překladač pokouší využívat paralelizace

a zřetězeného zpracování. Ještě lepších výsledků optimalizace lze dosáhnout pomocí ručního vkládání funkcí vytvořených v assembleru, případně užitím vnitřních funkcí překladače, tzv. *instrict funkcí*. Instrict funkce přímo odpovídají konkrétním instrukcím assembleru, ale narozdíl od nich jsou přenositelné i na jiný typ procesoru.

Výstupem překladače assembleru je přeložený strojový kód (*.obj) včetně přiřazených adres a správných rozsahů paměti. Linker sestaví z již vytvořených přeložených částí kódu jediný spustitelný soubor (*.out).

3.3.1 Práce v prostředí CCS

Na obr. 3.6 je zobrazeno vývojové prostředí CCS 3.1. Kromě hlavního editačního okna, kde je zobrazen zdrojový kód v jazyce C, zde vidíme standardní ovládací prvky. V levé části jsou umístěny informace o projektu a lze zde zobrazit a případně přidávat soubory související s projektem. Kromě hlavního zdrojového souboru zde vidíme inicializační a konfigurační soubory a rovněž používané knihovny funkcí.



Obr. 3.6. Ukázka prostředí Code Composer Studio 3.1

Zobrazený zdrojový kód realizuje jednoduchý program, který načte vstupy z AD převodníku, zesílí je a následně pošle zpět do DA převodníku. Po deklaraci potřebných proměnných následuje obsluha přerušení od kodeku a hlavní smyčka programu.

V hlavní smyčce se provede pouze inicializace vývojové desky a poté program skočí do nekonečné smyčky. Zpravování signálu realizuje kód obsluhy přerušení.

3.3.2 Vytvoření nového projektu v CCS

Po vytvoření projektu je nutné přidat soubor se zdrojovým kódem případně další soubory, ve kterých jsou umístěny použité funkce nebo konstanty. Potřebný je rovněž konfigurační soubor linkeru (*.cmd), ve kterém jsou definovány rozsahy paměťového prostoru procesoru. Ručně je dále potřeba přidat knihovny pro podporu DSP a periferií vývojové desky. V případě využití obsluhy přerušení je dále nutné ručně připojit soubor, ve kterém je uložen vektor přerušení od dané periferie. Naopak veškeré hlavičkové soubory (*.h) jsou do projektu načteny automaticky pomocí příkazu „*#include*“ ve zdrojovém kódu. Dále je vhodné nastavit možnosti kompilátoru a linkeru. Ovlivnit lze např. úroveň optimalizace zmíněné v kapitole 3.3. Pokud se veškeré potřebné hlavičkové soubory (*.h) a knihovny (*.lbr) nenachází v adresáři s projektem, je nutné zadat cesty, kde je má kompilátor hledat.

Alternativou, jak se vyhnout poměrně složitému nastavování periferií pomocí konfiguračních souborů a vektorů přerušení, je použití systému DSP/BIOS (viz kapitola 3.3.4).

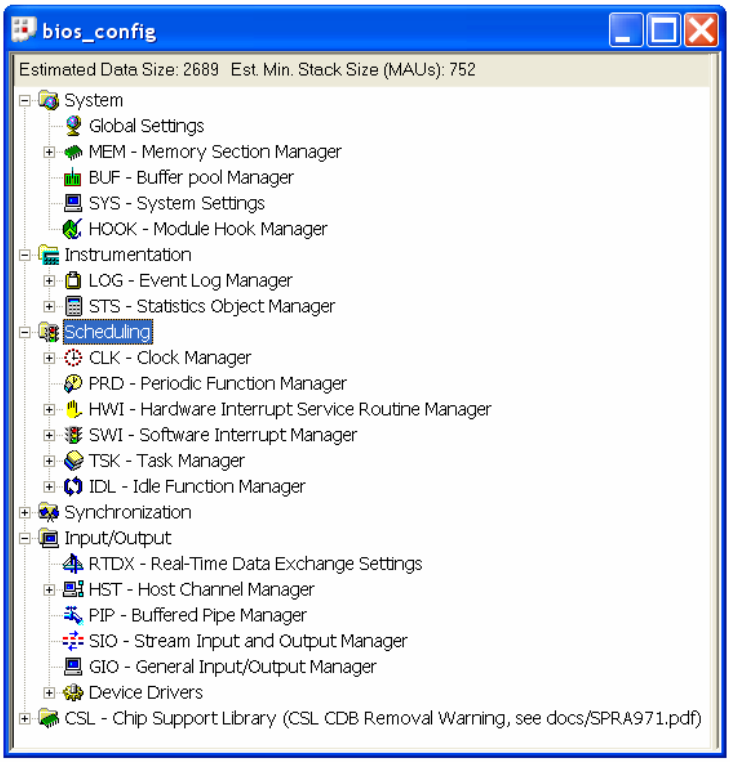
3.3.3 Ladění programu pro DSP v CCS

Pro ladění jsou k dispozici běžně dostupné funkce jako krokování spuštěného programu a umísťování breakpointů. Z prostředí CCS je také možné sledovat obsah libovolné proměnné, nebo přímo obsah registru či paměti. Na obr. 3.6 dole je sledována aktuální hodnota proměnné *volume*. Např. hodnoty signálu umístěného v bloku paměti lze zobrazit ve formě grafu, případně je možné nechat vykreslit diskrétní spektrum bloku signálu. Využít lze také programy napsané jazykem GEL (General Extension Language), pomocí kterého je možné vytvořit jednoduché testovací funkce nebo grafické prvky, kterými lze za chodu měnit parametry a hodnoty proměnných na straně DSP, v rámci prostředí CCS.

CCS dále umožňuje načíst vstupní data uložená v externím souboru z PC, což se hodí např. při testování algoritmů. Testovací data lze vygenerovat v Matlabu a poté je nechat zpracovat programem na připojené vývojové desce nebo v simulátoru DSP. Pokud již mámě shodný algoritmus odladěn v Matlabu, je možné oba výsledky porovnat, čímž zjistíme, zda program v DSP pracuje, tak jak jsme předpokládali.

3.3.4 Systém DSP/BIOS

DSP/BIOS představuje jádro operačního systému DSP pracující v reálném čase. Systém byl navržen pro úlohy, které vyžadují zpracování, plánování nebo synchronizaci v reálném čase a dále pro komunikaci s periferiemi. Pro chod využívá DSP/BIOS více vláken programu. Vlákno tvoří nezávislou posloupnost instrukcí, které neběží samostatně ale v rámci jiného programu. Běh jednotlivých vláken se automaticky střídá s využitím nuceného přerušování (tzv. preemptive multithreading), všechna vlákna také sdílí stejný adresový prostor.



Obr. 3.7. Konfigurace systému DSP/BIOS

Pro nastavení systému DSP/BIOS slouží konfigurační nástroj zobrazený na obr. 3.7. Pomocí tohoto nástroje lze zvolit, které periferie budou používány. Možné je definovat rozsahy pamětí, hardwarová (HWI) i softwarová (SWI) přerušení, případně periodické funkce (PRD), které se budou provádět periodicky v požadovaném časovém intervalu. Např. pro aktivaci HW přerušení zvolí programátor periferii a definuje symbol funkce, kterou poté zapíše do zdrojového kódu. Po uložení nastavení vytvoří konfigurační nástroj vlastní verzi operačního systému (firmware), který bude běžet v DSP. Výhodou systému DSP/BIOS je zejména jednoduchá konfigurace periferií a

snadné přizpůsobení požadované aplikaci. Nevýhodou jsou naopak zvýšené paměťové nároky, jelikož celý systém se musí načíst do paměti [8].

3.3.5 Systém RTDX

RTDX (Real Time Data eXchange) je technologie umožňující přenos dat v reálném čase mezi DSP a hostitelským PC. Při zpracování signálu je běžné, že DSP je aktivní jen po příchodu nového vstupního vzorku, ten je během krátké doby zpracován a poté činnost procesoru skončí. V tomto okamžiku může běžet systém RTDX.

RTDX tvoří součást vývojového prostředí CCS, které jej využívá ke komunikaci s připojeným DSP při ladění zavedeného programu. Avšak hostitelskou aplikací na straně PC může být i externí aplikace vytvořená v jazyku LabView, C++, Matlab nebo VisualBasic. Pro ovládání audio efektů byla v rámci této práce vytvořena aplikace s grafickým uživatelským rozhraním v Matlabu, podrobněji popsaná v kapitole 5.

4 Návrh a realizace algoritmů výsledných efektů

V této kapitole jsou popsány jednotlivé efekty od teoretického návrhu až po diferenční rovnice, pomocí kterých jsou efekty implementovány v DSP. Rovnice popisující efekty jsou pro názornost doplněny i v grafické podobě formou schémat. Matematický popis byl zvolen z důvodů vyšší přehlednosti, implementaci algoritmů v jazyku C pro DSP lze nalézt na přiloženém CD. Na CD jsou dále přiloženy i skripty v Matlabu, které byly použity v rámci návrhu jednotlivých efektů a také pro získání prezentačních dat a grafů obsažených v kapitolách 2 a 4.

Při číslicovém zpracování akustických signálů v reálném čase pomocí DSP lze zvolit mezi dvěma postupy. Zpracovávat je možné zvlášť každý nový vstupní vzorek nebo použít dávkového zpracování signálu za pomoci vstupního bufferu, který se nechá naplnit větším počtem vzorků a tento blok signálu se poté zpracovává najednou. Výhodou prvního postupu je minimální programové zpoždění (latence) mezi vstupem a výstupem, které má v tomto případě hodnotu vzorkovací periody. Např. při vzorkovací frekvenci 48 kHz činí zpoždění dané programem přibližně pouze 20 μs a zpracováván je skutečně vzorek po vzorku v reálném čase. Při dávkovém zpracování zpoždění roste s délkou signálových bufferů, avšak např. v algoritmech užívajících výpočet FFT je blokové zpracování nutností. Zpracování bloku vzorků ale také bývá při správné optimalizaci podstatně rychlejší, jelikož lze mnohem lépe uplatnit techniky paralelní zpracování dat.

Rovněž při zpracování zvuku pomocí PC není běžně umožněn přístup k jednotlivým vzorkům. Buffery se zavádí hlavně z toho důvodu, jelikož na PC běží (na rozdíl od systému se samostatným DSP) obvykle několik programů současně a nemáme tak zaručeno, zda bude v okamžik načtení vzorku dostatek systémových prostředků. Zpoždění signálu je menší problém, než když dojde k vynechání několika vzorků z důvodu, že se nestihly zpracovat. Problém se zpožděním nastává zejména u obyčejných zvukových karet, které pracují s velkými signálovými buffery a odezva mezi vstupem a výstupem je často velmi pomalá v hodnotách kolem 50 ms. Lépe jsou na tom profesionální zvukové karty pro hudebníky, u kterých si uživatel může zvolit velikost vyrovnávací paměti a tím odezvu dostatečně snížit. Běžně se tak dají nastavit hodnoty zpoždění v jednotkách ms, které již nejsme schopni sluchem rozeznat.

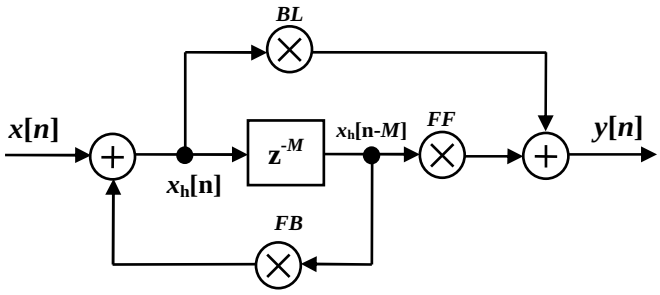
Vybrané efekty realizované v rámci této práci na DSP používají první zmíněnou metodu, zaměřenou na minimální odezvu. Případné vyšší zpoždění je tak zaváděno

zejména úmyslně (např. u ozvěnových efektů). Jisté zpoždění vzniká rovněž při filtraci signálu pomocí frekvenčních filtrů závislé na řádu filtru.

Na použitém DSP sice běží krom algoritmů pro zpracování audio signálů také méně náročný program pro ovládání efektů a komunikaci s PC, ten je ovšem v okamžiku načtení signálového vzorku hardwarově přerušen a poté se ihned vykoná algoritmus pro zpracování signálu (viz kapitola 5.3). Naprostá většina výpočetních prostředků DSP je tedy využita na zpracování zvukových vzorků.

4.1 Jednotka pro časově modulované efekty

Tato kapitola se zabývá návrhem jednotky pro efekty využívající modulaci zpoždění signálových vzorků. Základní skupinu efektů z této kategorie (delay, vibrato, flanger a chorus, viz kapitoly 2.2 a 2.3) realizuje architektura zobrazená na obrázku 4.1:



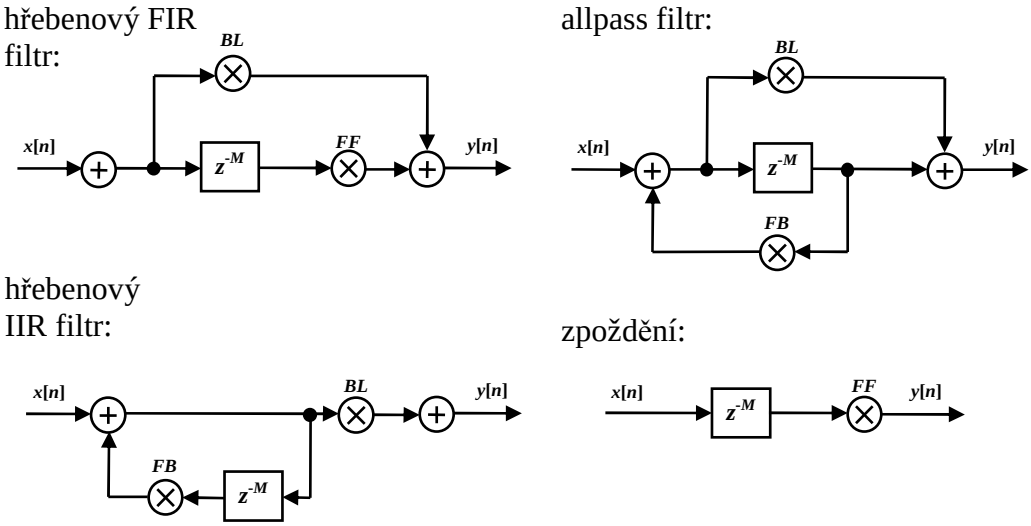
Obr. 4.1: Struktura realizující základní zpožďovací efekty

Schéma vychází ze struktury uvedené v [1]. Struktura vznikla modifikací hřebenových filtrů, které jsou popsány v kapitole 2.3 a dále z allpass filtru uvedeného v kapitole 2.4.1.

Architektura obsahuje zpožďovací člen, který realizuje zpoždění signálu o M vzorků, a celkem tří signálové větve, ve kterých lze nastavovat zesílení pomocí parametrů: *FF*, *FB* a *BL*. Parametr *FF* určuje zesílení zpožděných vzorků uložených v paměti, *BL* zesiluje přímou signálovou cestu ze vstupu na výstup a *FB* ovlivňuje zpětnou vazbu. Tabulka 4.1 spolu s obr. 4.2 shrnují funkce jednotky v závislosti na nastavených parametrech pro konstantní hodnotu zpoždění z^{-M}:

Tabulka 4.1: Souhrn funkcí jednotky realizující základní zpožďovací efekty:

funkce	popsáno v kap.	*BL*	*FB*	*FF*	podmínky stability
hřebenový filtr FIR	2.3	a	0	b	–
hřebenový filtr IIR	2.3	1	a	0	a < 1
allpass filtr	2.4	a	-a	1	–
zpoždění	2.2	0	0	1	–



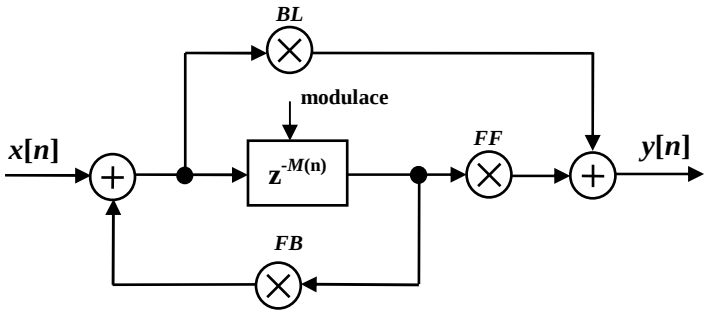
Obr. 4.2: Shrnutí funkcí základní efektové jednotky

Strukturu lze popsat pomocí následujících rovnic s využitím pomocné stavové proměnné x_h:

$$\begin{aligned} x_h &= x[n] + FB \cdot u[M] \\ y[n] &= FF \cdot u[M] + BL \cdot x_h \\ u[n] &= x_h \end{aligned} \tag{4.1}$$

Zpoždění je realizováno pomocí paměti tvořené polem (bufferem) *u*. Aktuální hodnoty proměnné x_h jsou ukládány do tohoto pole. První rovnice slouží pro výpočet pomocné stavové proměnné x_h, která je rovněž vyznačena ve schématu na obr. 4.1. Následuje výpočet výstupní hodnoty a dále uložení x_h do paměti.

Tabulka 4.1. shrnuje funkce pro konstantní hodnotu zpoždění *M*. Pro vytvoření dalších efektů (vibráto, flanger, chorus) je nutné zavést v čase proměnné zpoždění *M*(*n*), jehož hodnota je modulována vybraným průběhem, jak naznačuje obr. 4.3:



Obr. 4.3: Struktura realizující časově modulované efekty

Nejčastěji používanou modulací je sinusová funkce. Hodnota zpoždění $M(n)$ se přepočítává dle vztahu:

$$M(n) = 1 + d + w \cdot \sin(2\pi f_M \cdot n \cdot T_S), \tag{4.2}$$

kde d je počáteční hodnota zpoždění a w hloubka modulace (určující intenzitu efektu), f_M modulační frekvence a T_S vzorkovací perioda. Při implementaci algoritmu v reálném čase je nutné zabezpečit, aby hodnota $M(n)$ byla vždy kladná, jelikož nelze počítat s budoucími vzorky signálu (systém musí být kauzální). Hodnota zpoždění, počítaná dle vztahu 4.2, navíc nabývá i desetinných hodnot, avšak při odkazování na vzorek uložený v paměti musí být použit celočíselný index (ukazatel). Nejjednodušším řešením je hodnoty $M(n)$ zaokrouhlit, případně použít dolní/horní celou část. Skoková změna zpoždění vzniklá zaokrouhlováním má však negativní vliv na zvukovou kvalitu. Daleko lepším řešením je proto použití aproximace neceločíselného zpoždění, viz kapitola 4.3.1.

Výsledný efekt algoritmu závisí na modulační funkci a rovněž na parametrech určující přenos v jednotlivých signálových větvích (*FF*, *FB*, *BL*). Pomocí experimentálního nastavování modulací a koeficientů přenosů byla získána tabulka 4.2, která shrnuje příklady parametrů pro vytvoření různých efektů. Vhodným nastavením lze získat jak čisté tak kombinované efekty. Avšak hranice mezi čistými a kombinovanými efekty je často velmi těsná a značně subjektivní.

Tabulka 4.2: Příklady audio efektů v závislosti na parametrech:

funkce	parametry			zpoždění	modulace zpoždění		
	BL	*FB*	*FF*		typ	frekvence	hloubka
Vibráto	0	0,8	0	0–5 ms	sinus	3–8 Hz	0–2 ms
Flanger	0,6	0,6	0,6	0–0,5 ms	sinus	0,05–0,5 Hz	0–1,5 ms
Chorus (bílý)	0,9	0,3	-0,8	1–20 ms	NF šum(1)	–	0–5 ms
Chorus + Vibráto	0,9	0,7	0,3	0,02 ms	sinus	3 Hz	1 ms
Chorus (hluboký) + Vibráto	0,9	-0,3	0,6	16 ms	sinus	1,1 Hz	1,6 ms
Flanger (šum)	0,6	0,6	0,6	0,4 ms	NF šum(2)	–	0,02 ms

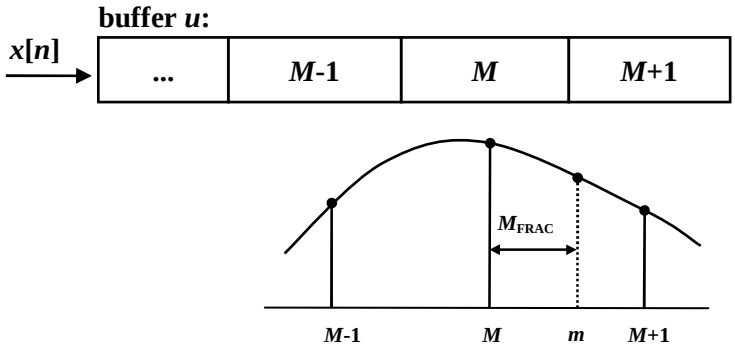
Pozn. 1: Vhodný je NF náhodný šum o frekvenčním rozsahu do 50 Hz

Pozn. 2: Náhodný šum o frekvenčním rozsahu do 20 Hz

Pro realizaci efektů využívajících modulaci bílým šumem (tab. 4.2) byla použita funkce `rand()`, která je součástí standardní knihovny funkcí jazyka C. Pro omezení frekvenčního pásma šumu byla užita přeladitelná dolní propust popsaná v kapitole 4.1.2. Volbou různých mezních frekvencí tohoto filtru byl zjištěn vliv frekvenčního rozsahu šumu na zvukový charakter efektů modulovaných šumem.

4.1.1 Implementace neceločíselného zpoždění vzorků

S využitím paměti číslicových systémů jsme schopni realizovat pouze zpoždění o hodnotách, které jsou rovny celočíselnému násobku vzorkovací periody. Důsledkem toho, že se zpoždění mění skokově a nikoliv plynule, vzniká zkreslení patrné zejména při pomalé změně hodnoty zpoždění (modulace signálem o nízké frekvenci). Tato nespojitost se ve výsledku projeví jako nežádoucí šum, což bylo prakticky vyzkoušeno na signálovém procesoru TMSC320C6416. Tento negativní jev lze významně eliminovat zavedením interpolace neceločíselného zpoždění vzorků, jak ilustruje obrázek 4.4:



Obr. 4.4: Princip interpolace neceločíselného zpoždění vzorků signálu

V horní části obrázku je naznačena paměť pro ukládání vstupních vzorků signálu. Index M reprezentuje ukazatel na hodnotu zpožděnou o M vzorků. V případě, kdy je aktuální zpoždění m neceločíselné, se tedy vypočítá lineární kombinace dvou nejbližších zpožděných vzorku podle vztahů:

$$\begin{aligned} M &= \lfloor m \rfloor \\ M_{FRAC} &= m - M \\ y[n] &= x[n-(M+1)] \cdot M_{FRAC} + x[n-M] \cdot (1 - M_{FRAC}) \end{aligned} \tag{4.3}$$

Nejprve se vypočítá hodnota M jako dolní celá část zpoždění m. M_{FRAC} tvoří desetinnou část zpoždění a $x[n]$ reprezentuje vektor uložených vstupních vzorků v paměti.

Ukázku zdrojového kódu algoritmu v jazyku C realizující časově modulované efekty lze nalézt v příloze B.

4.1.2 Návrh číslicové dolní propusti s IIR filtrem 2. řádu

Tato část kapitoly 4 se zabývá návrhem a odvozením vztahů pro výpočet koeficientů přeladitelného IIR filtru typu dolní propust s Butterworthovou aproximací přenosové funkce. Dolní propust je následně uplatněna pro omezení spektra náhodného

šumu, který se používá jako modulační signál k realizaci některých efektů uvedených v tabulce 4.2. Při použití náhodného šumu pro modulaci má hraniční frekvence dolní propusti funkci ovlivňující „rychlost modulace“. Odvozený filtr je dále uplatněn při realizaci efektu pohyblivé dolní propusti (kapitoly 2.5.3 a 4.3.3).

Návrh analogové dolní propusti:

Návrh číslicových filtrů IIR vychází z oblasti detailně popsaných analogových filtrů. Při návrhu všech typů Butterworthových filtrů (LP, BP, HP, atd.) se vyjde z normované dolní propusti, která je definovaná vztahem [3]:

$$\left|H^F(\omega)\right|^2 = \frac{1}{1+\left(\frac{\omega}{\omega_0}\right)^{2N}} \tag{4.4}$$

$H^F(\omega)$ je Fourierova transformace, N řád filtru, $\omega = 2\pi f$ úhlová frekvence a ω_0 mezní frekvence filtru, která je definovaná jako pokles $\left|H^F(\omega)\right|$ o 3 dB. Pomocí $\left|H^F(\omega)\right|^2$ je určen součin Laplaceových transformací filtru $H^L(s)H^L(-s)$. Dosazením za $s = j\omega$ získáme [3]:

$$H^L(s)H^L(-s) = \frac{1}{1+\left(\frac{s}{j\omega_0}\right)^{2N}} = \frac{1}{1-(-1)^N\left(\frac{s}{\omega_0}\right)^{2N}} \tag{4.5}$$

Pro póly kvadrátu přenosové funkce platí vztah [3]:

$$s_k = \omega_0 e^{j\frac{(N+1+2k)\pi}{2N}};\ 0 \le k \le 2N-1 \tag{4.6}$$

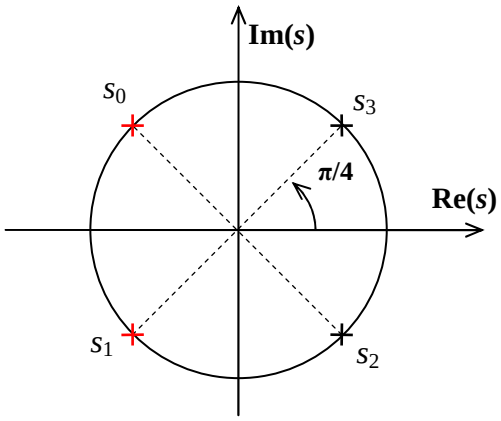
V případě filtru druhého řádu tedy stačí dosadit za $N = 2$:

$$s_k = \omega_0 e^{j\frac{(3+2k)\pi}{4}};\ 0 \le k \le 3 \tag{4.7}$$

Po dosazení za *k* získáme konkrétní póly:

$$s_0 = \omega_0 e^{j\frac{3\pi}{4}},\ s_1 = \omega_0 e^{j\frac{5\pi}{4}},\ s_2 = \omega_0 e^{j\frac{7\pi}{4}},\ s_3 = \omega_0 e^{j\frac{9\pi}{4}},$$

které lze zobrazit v komplexní *s*-rovině:



Obr. 4.5: Zobrazení pólů přenosu Butterworthova LP filtru 2. řádu v *s*-rovině.

Přičemž nás zajímá pouze prvních N pólů, v tomto případě s_0 a s_1, které leží v levé polorovině a splňují tak nutnou podmínku stability filtru.

Z pólů již lze získat přenosovou funkci požadovaného analogového filtru, dle následujícího vztahu [3]:

$$H^L(s) = \prod_{K=0}^{N-1} \frac{-s_K}{s - s_K} \tag{4.8}$$

Po dosazení vypočítaných pólů získáme:

$$H^L(s) = \frac{s_0 s_1}{(s - s_0)(s - s_1)}$$

$$H^L(s) = \frac{\omega_0^{\,2}}{\left(s + \omega_0 \frac{\sqrt{2}}{2}(-1 + j)\right)\left(s + \omega_0 \frac{\sqrt{2}}{2}(-1 - j)\right)} \tag{4.9}$$

Zjednodušením výrazu dostaneme přenosovou funkci v základním tvaru:

$$H^L(s) = \frac{\omega_0^{\,2}}{s^2 + s\sqrt{2}\omega_0 + \omega_0^{\,2}} \tag{4.10}$$

Digitalizace analogového filtru

Pro převedení analogového filtru na číslicový použijeme bilineární transformaci, definovanou substitucí, kde T_S je vzorkovací perioda:

$$s = \frac{2}{T_S} \cdot \frac{z - 1}{z + 1} \tag{4.11}$$

Po dosazení do rovnice 4.10 a několika úpravách dostaneme přenosovou funkci číslicového filtru definovanou *z*-transformací:

$$H(z)=\frac{\omega_0{}^2T_S{}^2\left(1+2z+z^2\right)}{z^2(4+2\omega_0\sqrt{2}T_S+\omega_0{}^2T_S{}^2)+z\left(-8+2\omega_0{}^2T_S{}^2\right)-2\sqrt{2}\omega_0T_S+\omega_0{}^2T_S{}^2+4} \quad (4.12)$$

Substitucí $\Theta_0=\omega_0T_S$ a vytknutím z^2 z čitatele i jmenovatele získáme obecný tvar (4.14) přenosové funkce:

$$H(z)=\frac{\left(1+2z+z^{-2}\right)\cdot\Theta_0{}^2}{(4+2\sqrt{2}\Theta_0+\Theta_0{}^2)+z^{-1}\left(-8+2\Theta_0{}^2\right)+z^{-2}\left(-2\sqrt{2}\Theta_0+\Theta_0{}^2+4\right)} \quad (4.13)$$

$$H(z)=\frac{b_0+b_1z^{-1}+b_2z^{-2}}{a_0+a_1z^{-1}+a_2z^{-2}} \quad (4.14)$$

Nyní již můžeme snadno vyjádřit vztahy pro výpočty potřebných koeficientů číslicového filtru:

$$\begin{aligned} a_0&=4+2\sqrt{2\Theta}+\Theta^2 & b_0&=\Theta^2\\ a_1&=2\Theta^2-8 & b_1&=2\Theta^2\\ a_0&=4-2\sqrt{2\Theta}+\Theta^2 & b_0&=\Theta^2 \end{aligned} \quad (4.15)$$

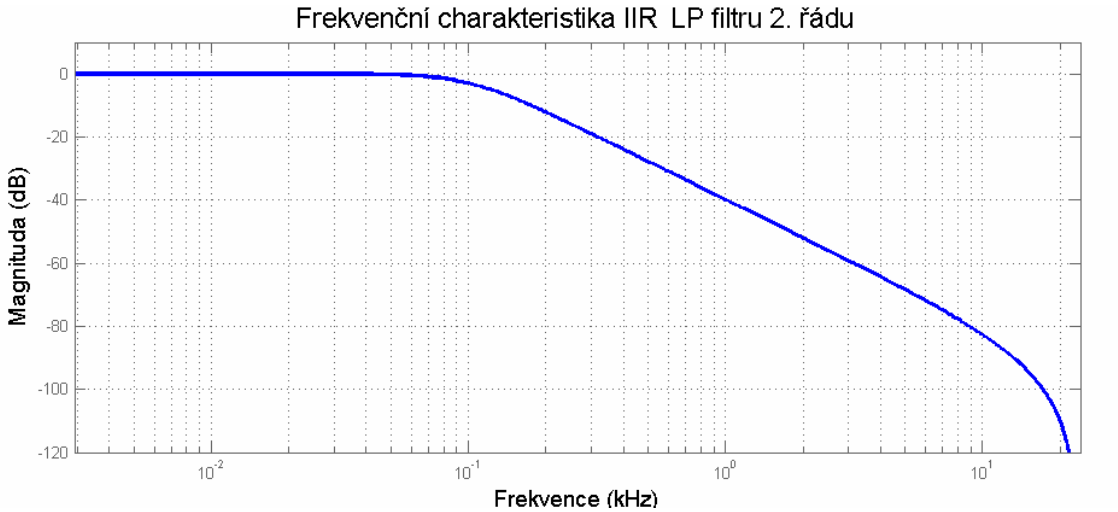
Všechny koeficienty dále převedeme do normalizovaného tvaru vydělením koeficientem a_0. Platí tedy $a_0=1$:

$$\begin{aligned} a_0&=1 & b_0&=\Theta^2/4+2\sqrt{2\Theta}+\Theta^2\\ a_1&=2\Theta^2-8 & b_1&=2b_0\\ a_0&=4-2\sqrt{2\Theta}+\Theta^2 & b_0&=b_0 \end{aligned} \quad (4.16)$$

Hodnotu Θ vypočteme na základě požadované mezní frekvence f_0 a dále podle zvolené vzorkovací frekvence f_S:

$$\Theta_0=\omega_0T_S=\frac{\omega_0}{f_S}=\frac{2\pi f_0}{f_S} \quad (4.17)$$

Na následujícím obrázku je zobrazen průběh frekvenční charakteristiky filtru, jehož koeficienty jsou vypočítány dle vztahů 4.16 pro mezní kmitočet $f_0=100\,\text{Hz}$ a vzorkovací kmitočet $f_S=48\,\text{kHz}$:



Obr. 4.6: Frekvenční char. Butterworthova LP filtru pro f_0 = 100 Hz a f_S = 48 kHz

4.1.3 Implementace IIR filtru v signálovém procesoru

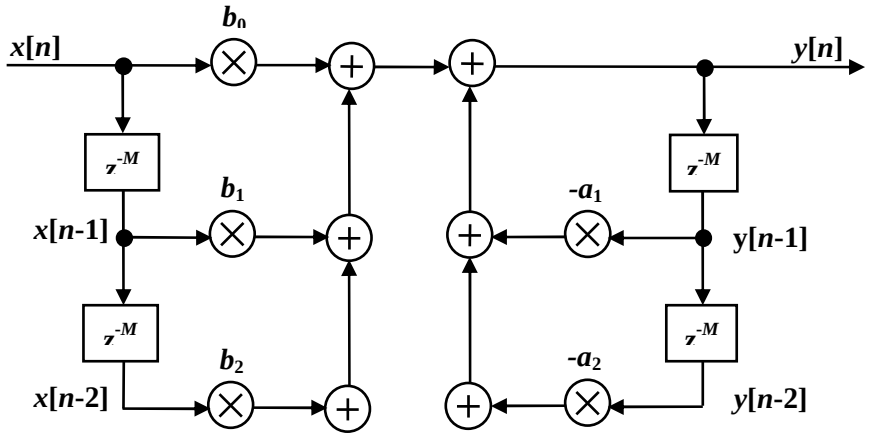
V kapitole 4.1.2 byl proveden teoretický návrh číslicových filtrů, jehož výsledkem jsou rovnice pro výpočet koeficientů filtru, zbývá tedy popsat realizaci filtračního algoritmu pro DSP. Základní způsob vychází přímo z diferenční rovnice systému. Diferenční rovnice obecného filtru IIR má tvar [2]:

$$y[n] = \sum_{i=0}^{M} b_i x[n-i] - \sum_{j=1}^{N} a_i y[n-j] \tag{4.18}$$

Pro filtr druhého řádu popsaného přenosovou funkcí 4.11 tedy platí:

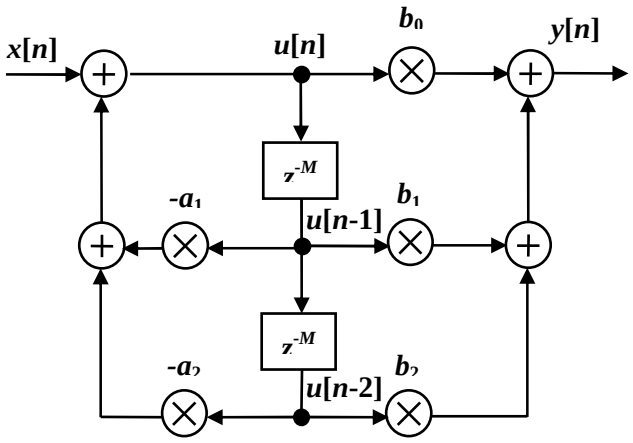
$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] - a_1 y[n-1] - a_2 y[n-2] \tag{4.19}$$

Podle předchozího rekurentního vztahu již lze jednoduše naprogramovat filtr v počítači nebo s využitím DSP. Tato struktura, zobrazená na obrázku 4.7, se nazývá *přímá struktura I* [3]:



Obr. 4.7: *Přímá struktura I* filtru IIR 2. řádu

Vidíme, že pro 2. řád filtru je potřeba čtyř zpožďovacích bloků, které jsou realizovány pomocí paměti. Přímou formu lze zjednodušit podle schématu níže (4.8). Tato struktura se označuje jako *přímá kanonická struktura II* [3]. Oproti schématu na obr. 4.7 je zde poloviční počet zpožďovacích bloků a výpočet tedy bude dvakrát méně náročný na kapacitu operační paměti.



Obr. 4.8: *Přímá kanonická struktura II* filtru IIR 2. řádu

Filtr lze popsat pomocí dvou rovnic:

$$\begin{aligned} u[n] &= x[n] - a_1 u[n-1] - a_2 u[n-2] \\ y[n] &= b_0 u[n] + b_1 u[n-1] + b_2 u[n-2] \end{aligned} \tag{4.20}$$

zavádíme zde pole pomocných stavových proměnných $u[n]$, které se ukládají do paměti (bufferu).

4.2 Stereofonní efekty

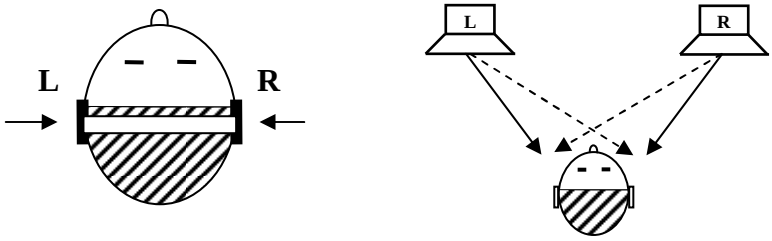
4.2.1 Stereo chorus, vibráto

Vztah 4.2 popisuje modulaci zpoždění sinusovou funkcí, která je vhodná pro realizaci efektů vibráto, chorus, nebo flanger. Velmi efektivního a hlubokého stereofonního vjemu lze dosáhnout, pokud zavedeme fázový posun v modulačních signálech mezi levým a pravým kanálem. Efekt bude nejvýraznější při posunu kanálů o 180°, které lze realizovat jednoduše otočením znaménka modulačního signálu u jednoho z kanálů. Modulace zpoždění pro levý a pravý kanál bude mít podobu:

$$\begin{aligned} M_L(n) &= 1 + d + w \cdot \sin(2\pi f_M \cdot n \cdot T_S) \\ M_R(n) &= 1 + d - w \cdot \sin(2\pi f_M \cdot n \cdot T_S) \end{aligned} \tag{4.21}$$

Nejlépe je tento efekt slyšitelný při poslechu na sluchátkách, kdy je každý kanál vnímán odděleně, viz obr. 4.9. Při aplikaci u efektu *flanger*, kde se používá velmi pomalá modulace zpoždění, působil na sluchátkách fázový posun mezi kanály o 180° až nepříjemně. Při poslechu na reproduktorech byl efekt vnímán jako plynulý pohyb zvuku mezi pravým a levým reproduktorem. Zejména při poslechu na sluchátkách se jeví jako vhodnější zavést mezi kanály menší fázový posun φ o hodnotách do 30°.

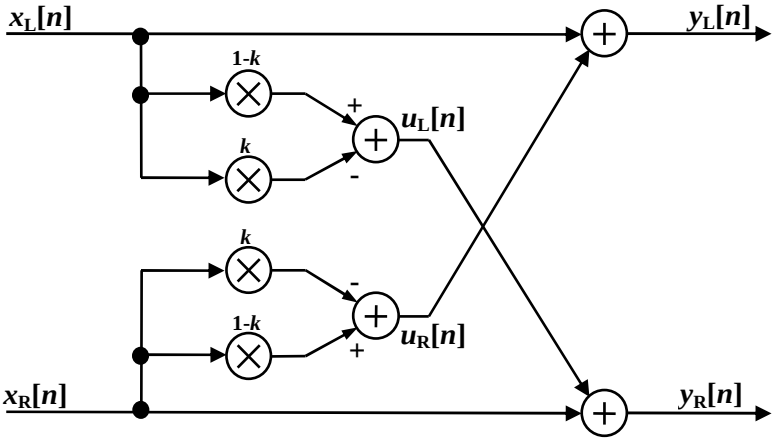
$$\begin{aligned} M_L(n) &= 1 + d + w \cdot \sin(2\pi f_M \cdot n \cdot T_S) \\ M_R(n) &= 1 + d + w \cdot \sin(2\pi f_M \cdot n \cdot T_S + \varphi) \end{aligned} \tag{4.22}$$



Obr. 4.9: Rozdíl stereofonního poslechu pomocí sluchátek a reproduktorů

4.2.2 Rozšíření stereofonní báze

Tzv. korektory stereofonní báze slouží ke zvýraznění případně potlačení prostorové informace mezi levým a pravým kanálem stereofonní nahrávky. Korektor znázorněný na obrázku 4.10 původně vychází z analogového obvodu tvořeného pomocí šesti operačních zesilovačů a dvojitého potenciometru [9].



Obr. 4.10: Korektor šířky stereofonní báze

Potenciometr je zde nahrazen bloky násobení s parametrem *k*. Parametr *k* slouží pro nastavení šířky efektu. Výstup prvního sumačního členu (signály u_L a u_R) je tvořen součtem přímého vstupního signálu s verzí téhož signálu s otočenou fází. V případě,

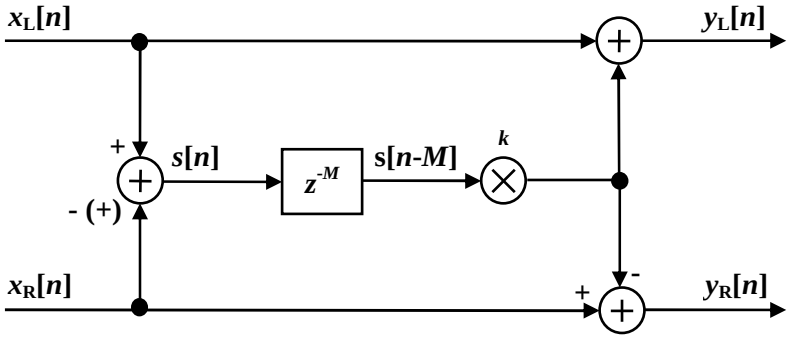
$k = 0{,}5$ se tyto složky vyruší a výstup levého i pravého kanálu zůstává nezměněn. Pro $k = 0$ vstupuje do součtu pouze přímý signál, který se následně smíchá s opačným kanálem. Oba výstupy tak nesou stejnou informaci a výsledkem je potlačení stereofonní informace. Rozšíření stereofonní báze naopak nastává při nastavení $1 \rangle k \rangle 0{,}5$, kdy se k oběma výstupům přičítá invertovaná varianta opačného kanálu. Při poslechu na stereofonní reprosoustavě se tak docílí efektu, jako kdyby se reproduktory vzdalovaly dále od sebe, při potlačování stereofonní informace zvuk naopak působí, jako kdyby reproduktory umístily blíže k sobě. Hodnoty $k \rangle 0{,}75$ jsou použitelné zejména pro signály s nižší stereofonní informací, pro běžné hudební nahrávky působí až příliš výrazně a dochází k roztržení stereofonní báze podobně, jako když se dvojice reproduktorů umístí vzhledem k posluchači příliš daleko od sebe.

Algoritmus lze popsat pomocí čtyř rovnic:

$$\begin{aligned} u_L[n] &= (1-k)\cdot x_L[n] - k \cdot x_L[n] \\ u_R[n] &= (1-k)\cdot x_R[n] - k \cdot x_R[n] \\ y_L[n] &= x_L[n] + u_R[n] \\ y_R[n] &= x_R[n] + u_L[n] \end{aligned} \tag{4.23}$$

4.2.3 Stereo reverb

Předchozí algoritmus, který neobsahuje žádné zpožďovací členy, lze snadno realizovat pomocí analogových obvodů. Zavedením zpoždění, viz obr. 4.11, lze dosáhnout rozmanitějších prostorových efektů. Rozdílem levého a pravého kanálu se získá stereofonní informace, která je následně zpožděna a přičtena k levému kanálu. K pravému kanálu se přičítá zpožděný signál s opačnou fází.



Obr. 4.11: Prostorový efekt využívající zpoždění signálu

Vliv hodnoty zpoždění na výsledný efekt shrnuje následující tabulka:

Tabulka 4.3: Vliv hodnoty zpoždění na efekt stereo korektoru:

hodnota zpoždění	popis efektu
< 5 ms	rozšíření stereofonní báze (viz korektor na obr. 4.10)
5 – 30 ms	prostorový dozvuk malé místnosti
30 – 150 ms	prostorová ozvěna, dozvuk větší místnosti
> 150 ms	dlouhá prostorová ozvěna, odrazy od skal

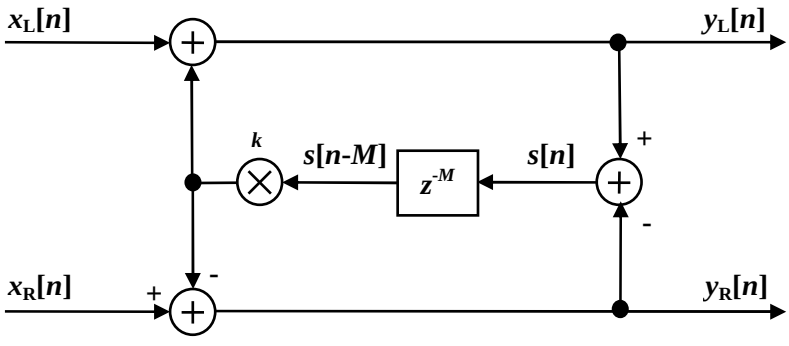
Pro malé hodnoty zpoždění (cca do 5 ms) je výsledný efekt velmi podobný jaký vytváří korektor zobrazený na obr. 4.10. Pro hodnoty přibližně od 5 do 30 ms je výsledný efekt velmi podobný dozvuku běžné místnosti. Zpoždění 30–150 ms již vytváří prostorovou ozvěnu a hodnoty vyšší než 150 ms působí jako stereo varianta ozvěny od dalekých překážek (viz *delay* – kapitola 2.2).

Intenzitu efektu lze ovlivnit parametrem k v použitelném rozsahu přibližně od 0 do 3. Pro $k = 0$ zůstávají oba kanály nezměněny. Algoritmus však upravuje pouze signál nesoucí stereofonní informaci, pokud však vstupní rozdílový člen nahradíme součtovým, bude efekt ovlivňovat i monofonní signály, viz kapitola 4.2.4. Strukturu lze realizovat pomocí následujících rovnic.

$$\begin{aligned} s[n] &= x_L[n] \pm x_R[n] \\ y_L[n] &= x_L[n] + k \cdot s[n-M] \\ y_R[n] &= x_R[n] - k \cdot s[n-M] \end{aligned} \tag{4.24}$$

Hodnota zpoždění M ovlivňuje velikost potřebné paměti (datové pole s).

Předchozí struktura je v podstatě varianta hřebenového filtru FIR (viz obr. 2.3) se zavedenou vazbou mezi levým a pravým kanálem. Podobným způsobem lze vytvořit i zpětnovazební algoritmus vycházející z hřebenového filtru IIR:



Obr. 4.12: Prostorový efekt využívající zpoždění signálu a zpětné vazby

Rozdíl je patrný zejména při zpožděních vyšších něž 30 ms, kdy se efekt projeví jako několikanásobný dozvuk případně ozvěna přecházející z jedné strany na druhou.

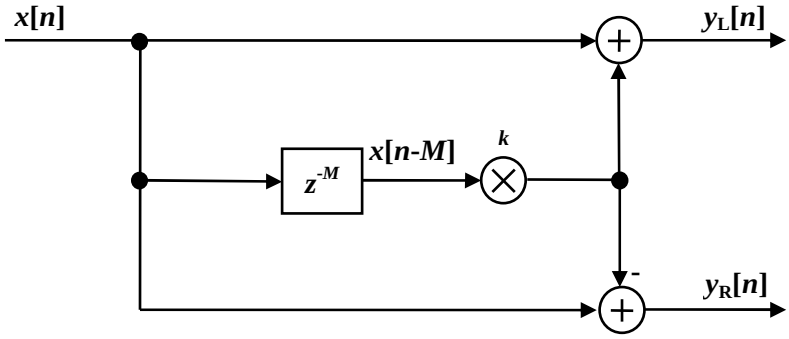
Intenzitu efektu lze opět ovlivnit koeficientem *k*, pro zajištění stability algoritmu pro libovolné vstupní signály je však nutné dodržet podmínku: $k \langle 0{,}5$.

Algoritmus lze opět přepsat pomocí rovnic:

$$\begin{aligned} y_L[n] &= x_L[n] + k \cdot s[n-M] \\ y_R[n] &= x_R[n] - k \cdot s[n-M] \\ s[n] &= y_L[n] \pm y_R[n] \end{aligned} \tag{4.25}$$

4.2.4 Vytvoření prostorového efektu u mono signálů

Zdrojem jednokanálového akustického signálu může být např. zpěvák nahrávaný jedním mikrofonem, případně elektrická kytara snímaná elektromagnetickými snímači. Avšak poslech monofonních hudebních signálů zní oproti stereofonním méně líbivě. Nabízí se proto možnost navodit u mono zvuku prostorový efekt. Nepatrnou úpravou stereo korektoru (obr. 4.11) lze získat algoritmus, který vytváří prostorový efekt u mono signálu, jak znázorňuje schéma 4.13:



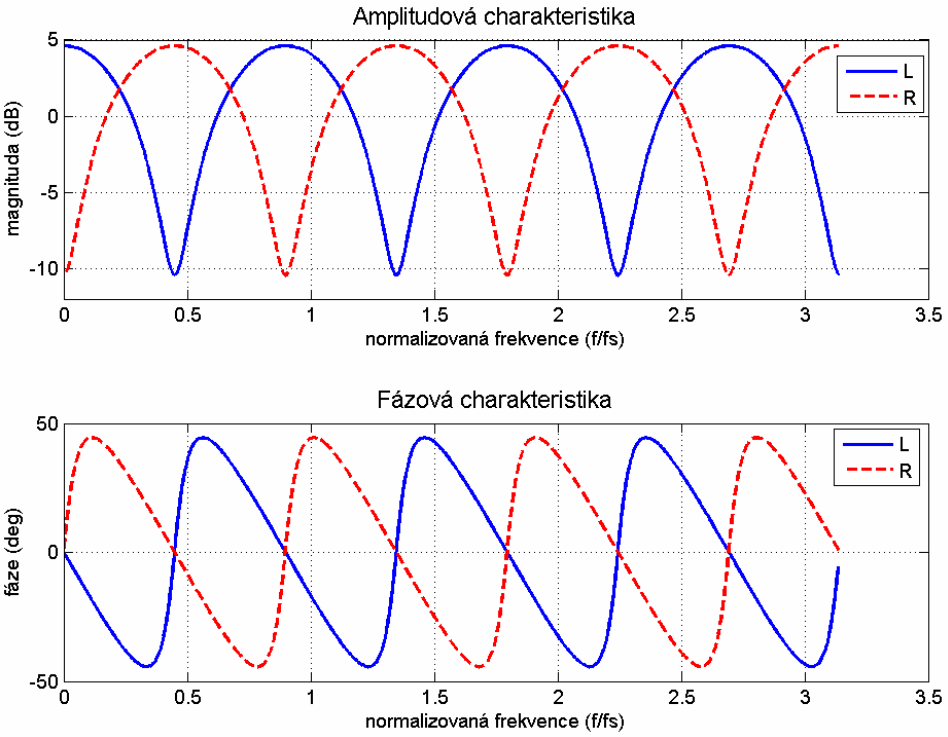
Obr. 4.13: Prostorový efekt u mono signálů

Schéma lze přepsat pomocí rovnic:

$$\begin{aligned} y_L[n] &= x[n] + k \cdot x[n-M] \\ y_R[n] &= x[n] - k \cdot x[n-M] \end{aligned} \tag{4.26}$$

Původní signál *x*[*n*] je nejprve zpožděn. Součet zpožděné varianty signálu s aktuálním signálem *x*[*n*] tvoří výstup levého kanálu a naopak rozdíl těchto signálů vytváří pravý kanál. Tato diference mezi vzniklými kanály vytváří jednoduchý, ale poměrně příjemný prostorový efekt. Intenzitu efektu lze ovlivnit koeficientem *k*. Změna hodnoty zpoždění má shodný vliv, jaký byl popsán u předchozího algoritmu (viz tabulka 4.3).

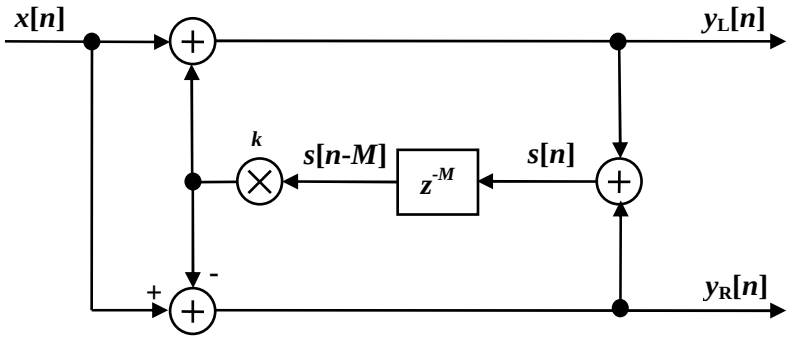
Rovnice 2.26 realizují v podstatě dva různé hřebenové FIR filtry, filtrující zvlášť levý a pravý kanál. Frekvenční amplitudovou a fázovou charakteristiku obou filtrů zobrazuje následující obrázek.



Obr. 4.14: Frekvenční char. dvou hřebenových FIR filtrů vytvářející stereo ozvěnu

Pro malé hodnoty zpoždění *M* lze pozorovat, že je každý kanál rozdělen na několik frekvenčních pásem a každé pásmo se u obou kanálů liší v amplitudě i ve fázi tak, že se navzájem doplňují. Mezi oběma kanály jsou tak vytvořeny diference ve fázi i ve složení spektra a tím je u posluchače vyvolán prostorový vjem.

Algoritmus popsaný z obr. 4.13. lze také snadno upravit zavedením zpětné vazby (dle obr. 4.15), čímž se získá postupně doznívající prostorová ozvěna.



Obr. 4.15: Prostorový efekt u mono signálu využívající zpoždění signálu a zpětné vazby

Schéma 4.15 lze realizovat pomocí rovnic 4.27:

$$\begin{aligned} y_L[n] &= x[n] + k \cdot s[n-M] \\ y_R[n] &= x[n] - k \cdot s[n-M] \\ s[n] &= y_L[n] + y_R[n] \end{aligned} \qquad (4.27)$$

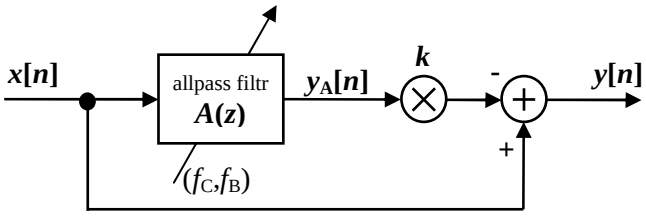
Opět je nutné dodržet podmínku stability: $k \langle 0{,}5$.

4.3 Efekty realizované pomocí laditelných filtrů

V této kapitole je popsána implementace efektů využívajících spektrální modulace pomocí laditelných filtrů.

4.3.1 Wah-wah

Základem efektu wah-wah popsaného v kapitole 2.5.1 je laditelný filtr typu pásmová propust. Pro sestavení dolní propusti je z hlediska výpočetní náročnosti výhodné použit allpass filtr druhého řádu popsaný v kapitole 2.4.1. Použitý algoritmus popisuje schéma 4.16:



Obr. 4.16: Realizace efektu wah-wah s BP filtrem sestaveným z allpass filtru 2. řádu

Samotný allpass filtr definovaný přenosovou funkcí 2.8 lze implementovat pomocí rovnice:

$$\begin{aligned} y_A[n] = &-c[n]\cdot x[n] + d[n](1-c[n])\cdot x[n-1] + x[n-2] \\ &- d[n](1-c[n])\cdot y_A[n-1] + cy_A[n-2] \end{aligned} \tag{4.28}$$

Pásmovou propust poté získáme rozdílem výstupu allpass filtru $y_A[n]$ se vstupním signálem $x[n]$ (viz přenosová funkce 2.9):

$$y[n] = x[n] - k\cdot y_A[n], \tag{4.29}$$

Činitelem k lze v rozsahu od 0 do 1 ovlivňovat intenzitu efektu (hloubku modulace). Pro hodnoty k větší než 0,6 je již efekt velmi výrazný a v některých případech může působit až nepříjemně. Koeficienty $c[n]$ a $d[n]$ v rovnici 4.28 jsou přepočítávány v čase podle aktuální hodnoty mezní frekvence filtru $f_C[n]$ dle rovnic 4.30:

$$\begin{aligned} c[n] &= \frac{\tan(\pi\cdot f_B[n]/f_S) - 1}{\tan(2\pi\cdot f_B[n]/f_S) + 1} \\ d[n] &= -\cos(2\pi\cdot f_C[n]/f_S) \end{aligned} \tag{4.30}$$

Mezní frekvence filtru se v případě zvolené varianty efektu wah-wah mění periodicky:

$$f_C[n] = f_{MIN} + 0{,}5\, f_{MAX}\cdot(1+\sin(2\pi\cdot rate\cdot n\cdot T_S)) \tag{4.31}$$

f_{MIN} a f_{MAX} definují frekvenční rozsah, ve kterém se bude filtr přelaďovat, parametrem *rate* nastavujeme rychlost „kvákadla“ a T_S značí vzorkovací periodu. Šířka pásma filtru

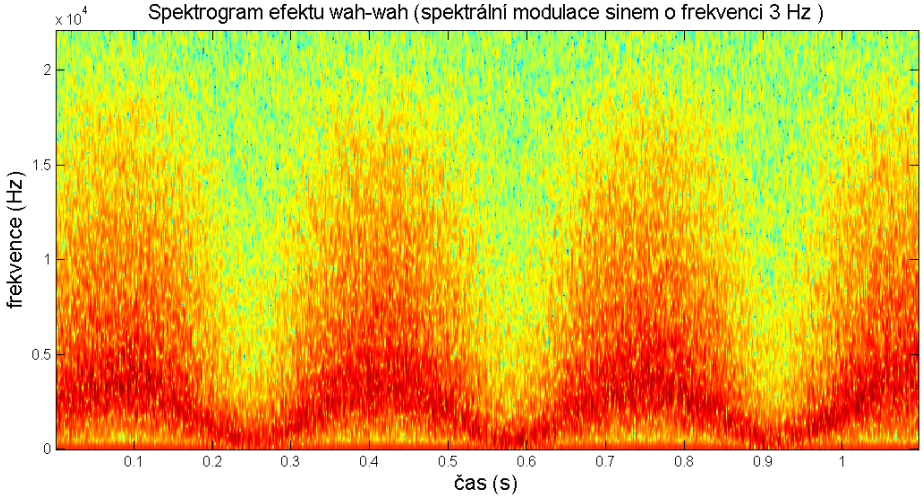
f_B je přepočítávána tak, aby byl poměr hodnoty mezní frekvence ku šířce pásma konstantní:

$$f_B[n] = f_C[n] \cdot width \tag{4.30}$$

Parametr *width* reprezentuje relativní šířku pásma a ovlivňuje tak výsledný charakter efektu (vhodné hodnoty jsou od 0 do 0,5).

Pro zachování jednotkového zesílení v propustném pásmu by se výstup (vztah 4.29) měl dle přenosové funkce 2.9 navíc vydělit dvěma, avšak pro zachování přibližně stejné hlasitosti zpracovávaného signálu je dvojnásobné zesílení vhodné, neboť kompenzuje útlum signálu způsobený pohyblivou pásmovou propustí.

Vliv efektu wah-wah je možné ilustrovat při aplikaci na bílý šum pomocí spektrogramu:

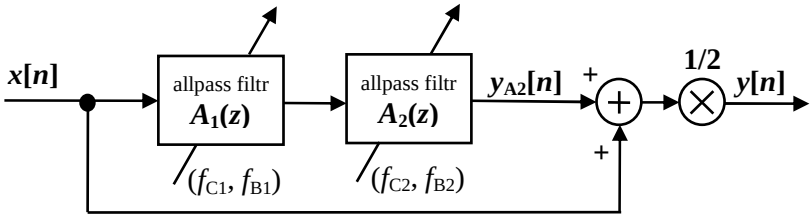


Obr. 4.17: Spektrogram bílého šumu zpracovaného efektem wah-wah

Tmavá místa zobrazují propustné pásmo, naopak světlejší barvou jsou znázorněny potlačené frekvence. Vidíme, že pásmová propust se periodicky přelaďuje.

4.3.2 Phaser

Nahradíme-li pásmovou propust u efektu wah-wah pásmovou zádrží, získáme *Phaser*, popsány v kapitole 2.5.2. Pásmovou zádrž vytvoříme rovněž pomocí allpass filtru (viz vztahy 2.8 a 2.9) dle následujícího schématu:



Obr. 4.18: Realizace efektu phaser pomocí dvojice přeladitelných pásmovou propustí

Algoritmus se od pásmové propusti liší pouze ve znaménku u sumačního členu.

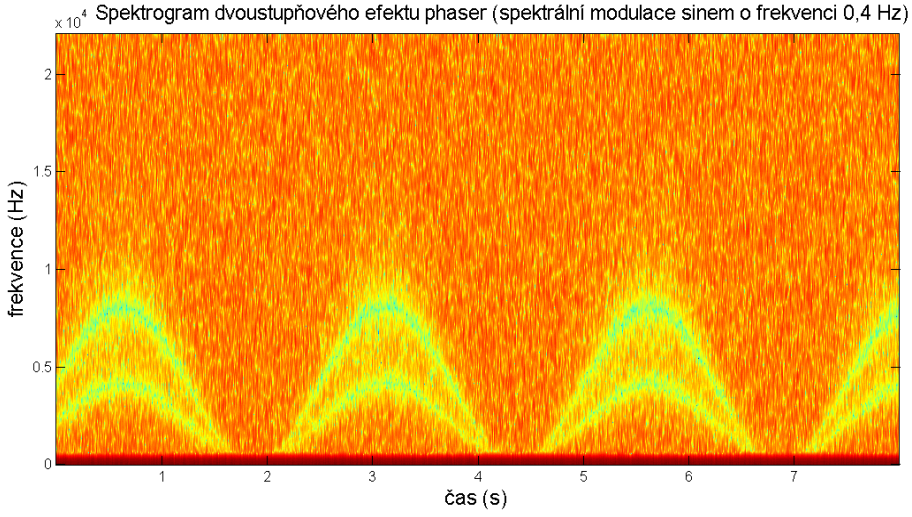
Užití pouze jednoho filtru druhého řádu se ukázalo jako nedostatečné a výsledný efekt nebyl příliš výrazný. Realizace pomocí dvojice kaskádně řazených filtrů již poskytuje lepší výsledek. Obě pásmové propusti lze přelaďovat shodně, čímž se zvýší strmost filtru. Mezní frekvence je také možné ladit s určitým frekvenčním rozestupem, čímž se získá lehce odlišný výsledný efekt. V takovém případě ovšem stoupne výpočetní náročnost, jelikož je nutné přepočítávat dvojnásobný počet koeficientů pro oba filtry zvlášť.

Pro přepočet mezních frekvencí f_B, f_C lze opět použít rovnice 4.31 a pro koeficienty filtru platí vztahy 4.30. Filtrace se provede dvakrát dle vztahu 4.28 a získaný signál y_{A2} se poté sečte se vstupem.

$$y[n] = 0.5(x[n] + k \cdot y_{A2}[n]) \tag{4.32}$$

Útlum úzké pásmové zádrže nemá tak výrazný vliv na celkovou hlasitost signálu jako v případě efektu wah-wah, proto zde zůstává násobení hodnotou 1/2 dle definice filtru přenosovou funkcí 2.9. Intenzitu efektu lze ovlivňovat změnou šířky pásma filtrů .

Účinek efektu phaser je možné rovněž ilustrovat spektrogramem, kde byl jako vstupní signál zvolen bílý šum (obr. 4.19)

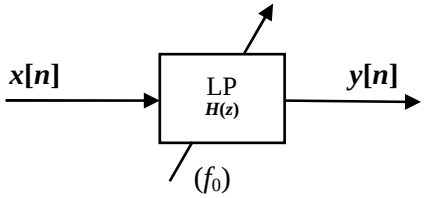


Obr. 4.19: Spektrogram bílého šumu zpracovaného efektem phaser

Útlum ve frekvenčním pásmu znázorňují světlá místa spektrogramu. Vidíme zde vliv dvou pásmových zádrží, které se v čase přelaďují se stejnou periodou ale v různých frekvenčních rozsazích.

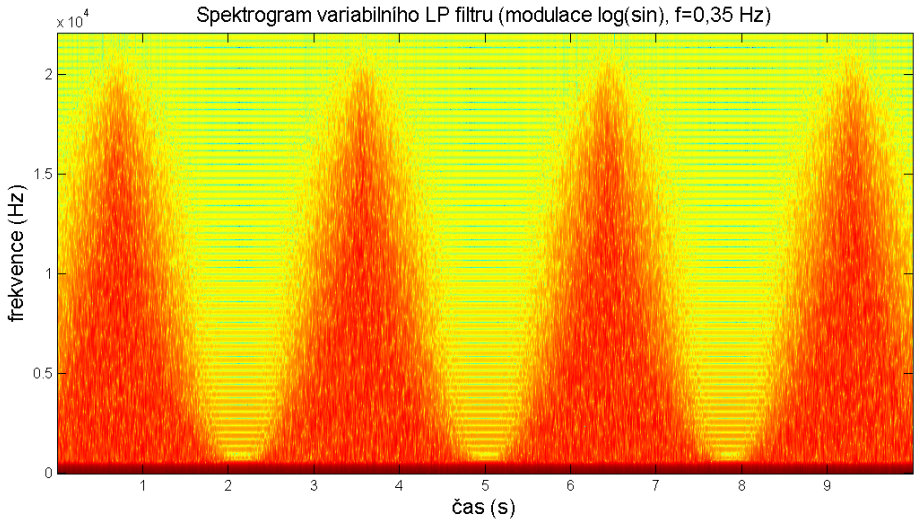
4.3.3 Variabilní dolní propust

Jako laditelná dolní propust je použit Butterworthův filtr 2. řádu, který byl již popsán v kapitole 4.1.2. Pro změnu mezní frekvence je nutné přepočítávat koeficienty (dle vztahů 4.16), jež byly odvozeny ve stejné kapitole. Pro implementaci filtru byla použita *Přímá kanonická struktura II*, definovaná v kapitole 4.1.3.



Obr. 4.20: Variabilní dolní propust

Následující obrázek zobrazuje spektrogram dolní propusti, která se přelaďuje podle periodického průběhu s frekvencí 0,35 Hz v rozsahu 100 Hz až 22 kHz. Tmavá místa reprezentují opět propustné pásmo.



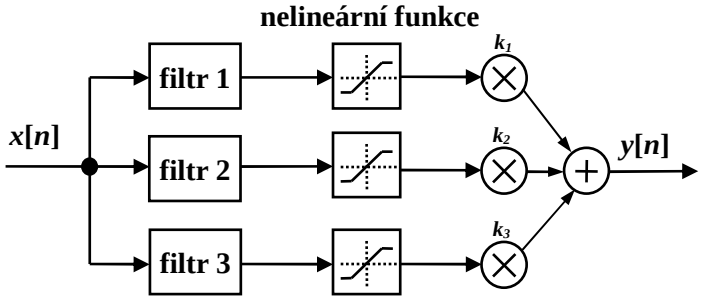
Obr. 4.21: Spektrogram bílého šumu zpracovaného variabilním LP filtrem

4.4 Jednotka pro realizaci efektů zkreslení

V kapitole 2.6.1 jsou popsány základní jednoduché typy zkreslení elektrické kytary. S rozvojem rockové a metalové hudby však vznikla řada komplexnějších kytarových efektů, vytvářející velmi silné zkreslení se specifickým „tvrdým“ zvukem.

Většina světových výrobců efektových pedálů, jako např. japonská firma Boss, nabízí vlastní patentované typy zkreslení, které vytváří charakteristický zvuk, známý u hudebníků po celém světě. Cílem této práce však bylo vytvořit vlastní parametrickou efektovou jednotku, která by umožnila realizovat širší škálu zkreslujících efektů.

Výsledný zvuk zkreslené kytary lze částečně ovlivňovat nastavením asymetrických prahů pro zkreslení. Další možností je zařazení ekvalizéru na výstup. Ekvalizérem lze měnit amplitudy jednotlivých harmonických složek signálu, avšak není možné ovlivňovat vznik nových harmonických. Následující myšlenka spočívá v rozdělení vstupního signálu do několika frekvenčních pásem [1] (které se případně i překrývají) a následná aplikace nelineárních funkcí, jak naznačuje následující obrázek:



Obr. 4.22: Vícepásmové zpracování signálu

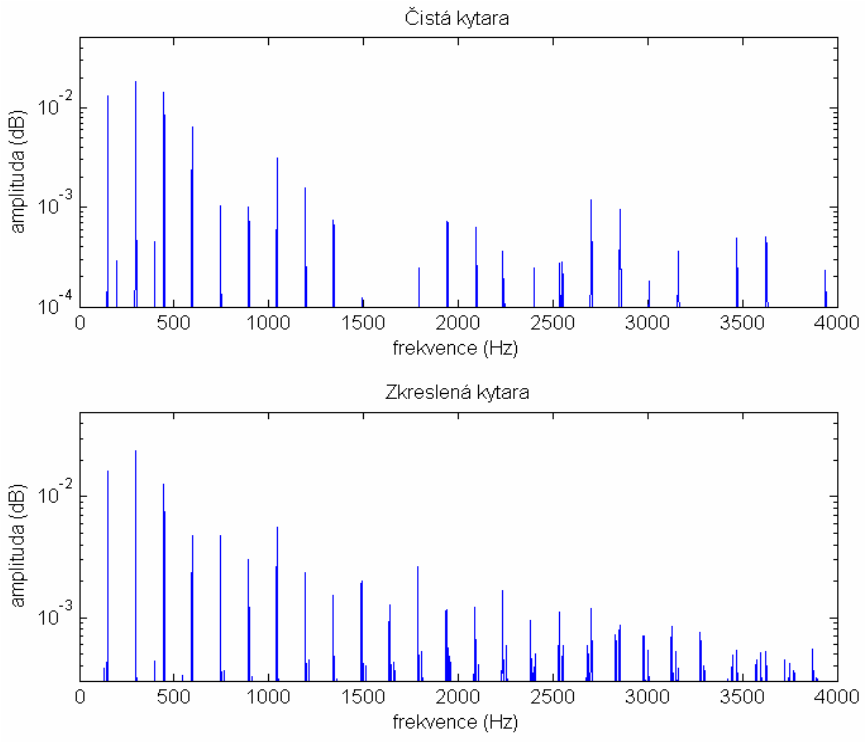
Po rozdělení signálu se v každé větvi provede konkrétní nelineární funkce a výstupy se s různou vahou opět sloučí. Vhodným filtrem, který lze zařadit před zkreslující funkci, je dolní propust, jelikož zachovává první harmonickou složku určující výšku tónu.

4.4.1 Vliv dolní propusti na charakter zvuku zkreslené kytary

Struna upevněná na dvou koncích kytary představuje soustavu kmitajících bodů, na které vzniká stojaté vlnění. Půlvlna v celé délce struny vytváří základní harmonický tón o frekvenci *f*. Kromě toho mohou vznikat další dvě, tři, čtyři, atd. kratší půlvlny, jejichž frekvence odpovídá násobkům základní frekvence 2*f*, 3*f*, 4*f*, atd. [6]. Počet vyšších harmonických a jejich amplituda závisí na provedení strunného nástroje nebo na bodě, ve kterém dojde k rozkmitání struny. Při zavedení zkreslení kytarového signálu tak vznikají nové harmonické frekvence odvozené od frekvenčních složek čistého tónu kytary.

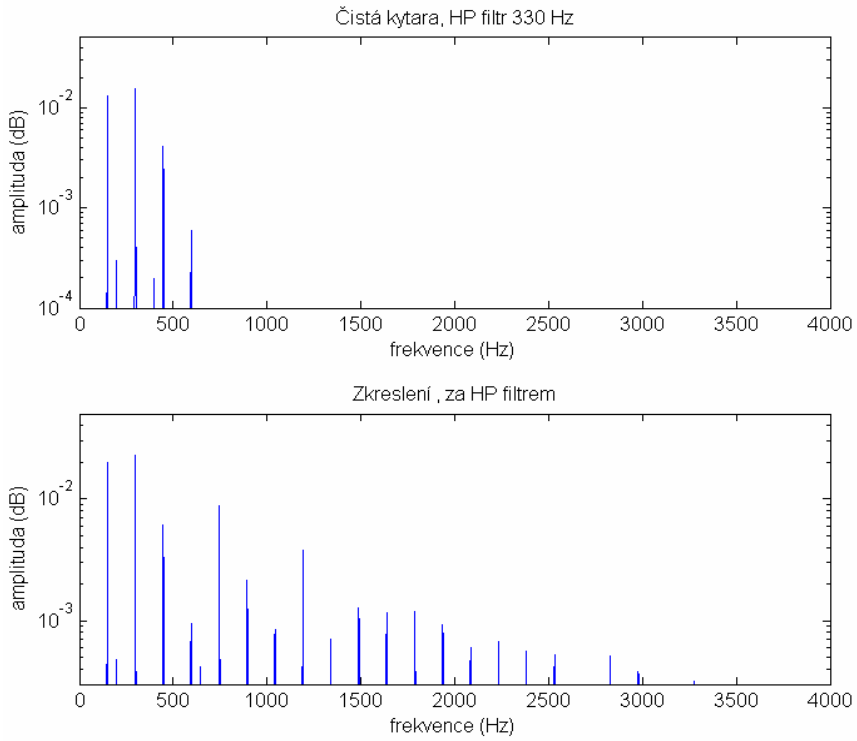
Většina modelů běžných elektrických kytar je vybavena výstupní laditelnou dolní propustí. Tímto filtrem lze potlačit vyšší frekvenční složky v původním nezkresleném signálu, čímž se ovlivní i vznik nových harmonických složek po zavedení zkreslení.

Obr. 4.23 zobrazuje frekvenční složení čistého tónu *d* (146,83 Hz) elektrické kytary a dále zkreslenou variantu téhož signálu pomocí asymetrické limitace (viz kapitola 2.6.1):



Obr. 4.23: Frekvenční průběh čisté a zkreslené kytary, struna d (147 Hz)

Následující obrázek zobrazuje stejný signál filtrovaný dolní propustí 4. řádu o mezním kmitočtu 330 Hz a pod ním je opět zkreslená varianta.



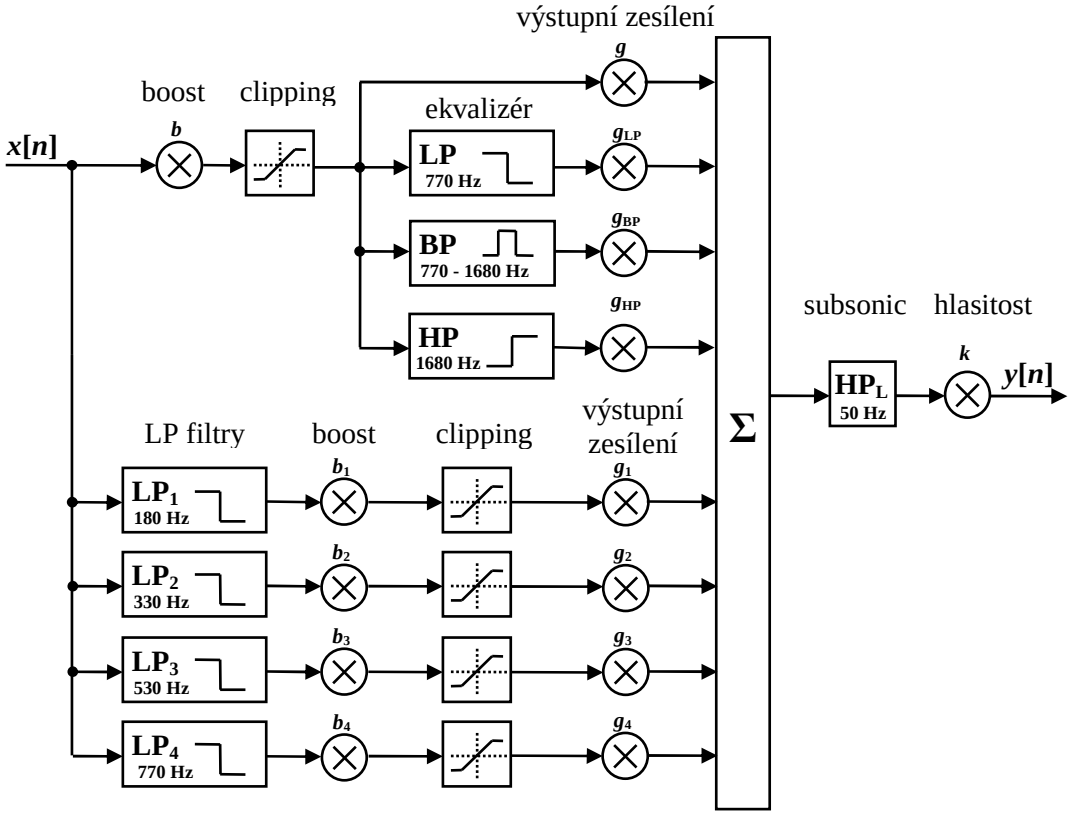
Obr. 4.24: Frekvenční průběh čisté a zkreslené kytary upravené LP filtrem, struna d (147 Hz)

Zařazením LP filtru před nelineární funkci lze tedy ovlivňovat vznik nových (zejména nižších) harmonických frekvencí a měnit tak výslednou barvu tónu jiným způsobem než pomocí ekvalizéru.

4.4.2 Struktura jednotky realizující efekty zkreslení

Vlastní navržený efekt se skládá ze dvou signálových větví. V horní části obr. 4.25 se nachází větev, ve které se signál nejprve zkreslí (blok *clipping*). Míra zkreslení se nastavuje vybuzení vstupního zesílení *b* (*boost*). Po zkreslení následuje ekvalizér, kterým lze dále ovlivnit charakter zvuku. Druhá část se skládá ze čtyř dolních propustí umístěných na vstupu (obr. 4.25 dole), které jsou následovány bloky realizující zkreslení. Intenzitu zkreslení jednotlivých pásem lze opět měnit (parametry b_1–b_4).

Výstup je tvořen váhovým součtem všech signálových větví. Vhodným nastavením parametrů g_i tak lze nastavit poměrně různorodé typy kytarového zkreslení (viz tabulka 4.4). Menší vliv na výsledný zvuk má rovněž volba prahů limitace. Výstup je dále doplněn subsonickým filtrem, který odstraní stejnosměrnou složku a nežádoucí nízké frekvence a dále výstupním zesílením *k*, který určuje hlasitost.



Obr. 4.25: Jednotka realizující kytarové efekty zkreslení

Tabulka 4.4: Vliv výstupních parametrů na zvukový charakter efektu:

výstupní parametry	ovlivňuje frekvence
g_{LP}, g_1, g_2	nižší harmonické
g_{BP}, g_3, g_4	střední kmitočty
g_{HP}	vyšší harmonické
g	všechny kmitočty

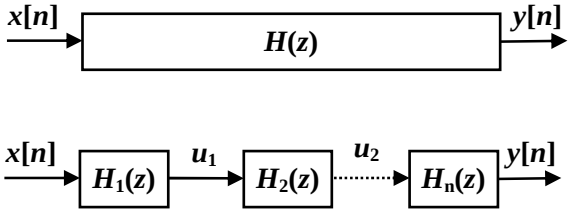
Výstup jednotky lze doplnit efekty jako flanger, chorus, phaser, které byly popsány výše v kapitole 4, čímž lze dosáhnout ještě rozmanitějších úprav ve výsledném zvuku.

Veškeré filtry jsou navržené dle Butterworthovy aproximace. Jako optimální se ukázalo použití filtrů 4. řadu, které nabízejí dostatečnou strmost při relativně nízkých HW nárocích. Mezní frekvence filtrů byly zvoleny na základě experimentů a poslechu výsledného zvuku. Implementace filtrů v signálovém procesoru je provedena pomocí kaskádního řazení *SOS filtrů* popsaných v následující kapitole.

Jelikož efekt vytváří vysoký počet vyšších harmonických, hodí se zejména pro výrazné zkreslení při hře na jednu strunu nebo jednodušších akordů. V případě složitějších akordů nemusí výsledek působit příjemně, jelikož vyšší harmonické tóny již nejsou v harmonii (viz kapitola 1.1.4). Pro vyšší kytarové tóny také nelze vyloučit vliv zkreslení vniklého aliasingem zejména v signálové větvi, která nezahrnuje vstupní LP filtr.

4.4.3 Implementace IIR filtrů vyšších řádů

Při realizaci diskrétního systému jsou koeficienty diferenční rovnice reprezentovány ve dvojkové soustavě s konečnou přesností. Tato nepřesnost má vliv i na stabilitu filtračních algoritmů. Pro minimalizaci vlivu kvantování na číslicové filtry je vhodnější použít místo filtrů vysokých řádů přímých struktur filtry kaskádních nebo paralelních struktur [8]. Při implementaci IIR filtrů vyšších řádů byly použity kaskádní filtry složené z IIR filtrů druhého řádu. tzv. „*Second Order Section*“ neboli *SOS* filtry. Princip kaskádní struktury *SOS* zachycuje obrázek 4.26. Filtr sudého řádu popsaný přenosovou funkcí $H(z)$ lze rozložit na několik kaskádně řazených filtrů druhého řádu.



Obr. 4.26: Přímá a kaskádní struktura filtru IIR

Obecný IIR filtr je popsán přenosovou funkcí 4.33 [2]. Čitatel i jmenovatel se dají rozložit na součin polynomů druhého stupně podle vztahu 4.34 [2]. Koeficienty jednotlivých součinů v čitateli a jmenovateli odpovídají koeficientům *SOS* filtru.

$$H[z] = \frac{\sum_{i=0}^{M} b_i z^{-i}}{1 + \sum_{j=1}^{N} a_j z^{-j}} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + ... + b_M z^{-M}}{1 + a_1 z^{-1} + a_2 z^{-2} + ... + a_N z^{-N}} \tag{4.33}$$

$$H[z] = C \prod_{i=1}^{N/2} \frac{b_{0i} + b_{1i} z^{-1} + b_{2i} z^{-2}}{1 + a_{1i} z^{-1} + a_{2i} z^{-2}} \tag{4.34}$$

Pro návrh Butterworthova filtru vyššího řádu lze použít funkci v programu Matlab:

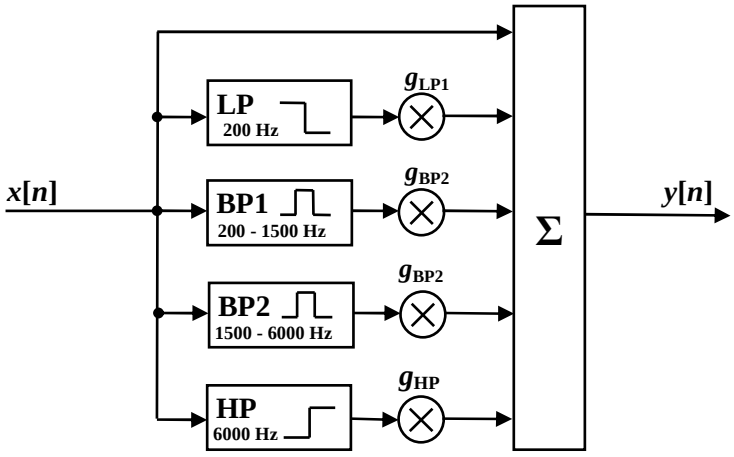
`[b,a,k] = butter(n,2fc/fs,'ftype')`,

kde `a` a `b` jsou koeficienty FIR filtru, k koeficient zesílení, `n` je řád filtru `fc` mezní frekvence a `fs` vzorkovací frekvence. Následuje definice typu filtru: HP: `'high'`, LP: `'low'`, BP: `'bandpass'`. Přepočet koeficientů z přímé struktury IIR filtru na *SOS* filtr lze v Matlabu provézt pomocí funkce, která vrací matici *SOS* koeficientů:

```
[sos]=tf2sos(b,a,k);
```

4.5 Ekvalizér a ovládání hlasitosti.

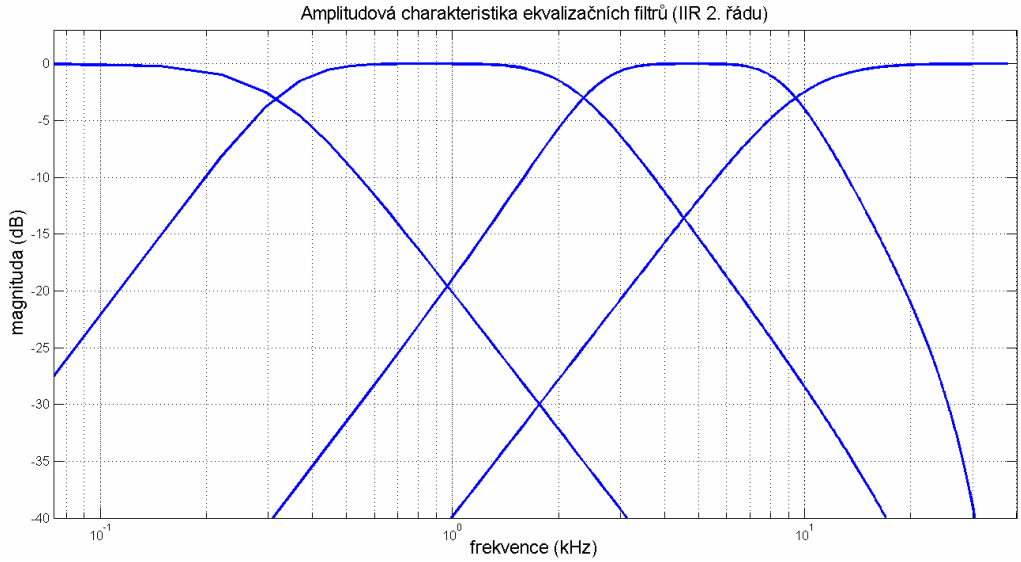
Kromě specifického ekvalizéru, který je součástí jednotky realizující efekty zkreslení, byl dále vytvořen samostatný 4-pásmový ekvalizér, kterým je možné doplňovat i ostatní efekty. Strukturu ekvalizéru zobrazuje následující obrázek:



Obr. 4.27: Realizace 4-pásmového ekvalizéru

Ekvalizér se skládá ze čtyř Butterworthových filtrů IIR 2. řádu v DSP realizovaných dle rovnic 4.20. Dolní propust (LP) slouží pro korekci basů, první pásmová propust (BP1) realizuje pásmo středních kmitočtů, druhá pásmová propust (BP2) vyšší střední kmitočty. Horní propust (HP) ovlivňuje výšky. Pomocí parametrů g_i lze posílit, případně potlačit jednotlivá pásma.

Pásma jednotlivých filtrů jsou zvolena s ohledem na nelineární vlastnosti lidského sluchu. Frekvenční amplitudové charakteristiky jednotlivých filtrů zobrazuje následující obrázek:



Obr. 4.28: Amplitudové charakteristiky ekvalizačních filtrů

4.5.1 Ovládání hlasitosti

Pro nastavení úrovní jednotlivých pásem ekvalizéru nebo regulaci vstupní a výstupní hlasitosti je vhodné použít hodnoty v decibelech (viz kapitola 1.2.3). Přepočet mezi dB a lineárními hodnotami definují vztahy:

$$g_{dB} = 20\log(g), \tag{4.35}$$

$$g = 10^{\left(\frac{g_{dB}}{20}\right)}, \tag{4.36}$$

kde logaritmus je o základu 10. Např. regulační rozsah ±12 dB umožňuje přibližně čtyřnásobné zesílení či zeslabení signálu.

4.6 Signalizace úrovně signálu

Při převodu číslicového signálu na analogový je použit 16b DA převodník, kterým je vybavena vývojová deska C6416 DSK (viz kapitola 3.2). DA převodník je

schopen zpracovat hodnoty v rozsahu od -32,768 do 32,767. V případě, kdy je do DA převodníku poslána vyšší hodnota, dojde k limitaci a nežádoucímu velmi nepříjemnému zkreslení signálu. Proto je vhodné sledovat úroveň výstupního signálu.

K realizaci indikátoru úrovně signálu jsou využity čtyři LED, které jsou součástí vývojové desky C6416 DSK. Prahy indikace jednotlivých LED shrnuje tabulka 4.5:

Tabulka 4.5: LED indikátor úrovně signálu:

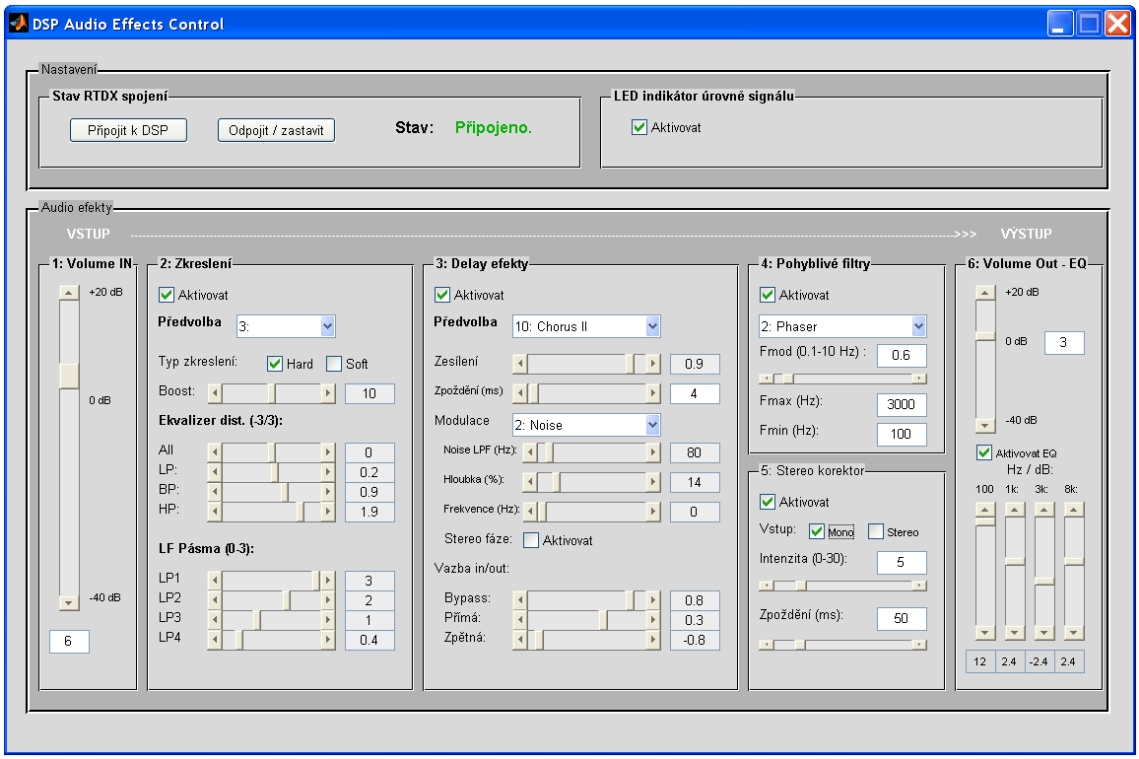
LED č.	úroveň (dB)	úroveň (16b integer)
1	-24	4100
2	-12	8200
3	-6	16400
4	-3	23200

5 Realizace ovládání efektů a komunikace DSP s PC

Vývojová deska C6416 je vybavena čtyřmi spínači, které je možné využít pro základní ovládání programu DSP. Spínače lze naprogramovat např. pro přepínání mezi předem připravenou sadou efektů s různými parametry. Avšak pro plnou kontrolu nad realizovanými algoritmy v reálném čase bylo vhodnější vytvořit grafickou PC aplikaci, která by komunikovala s vývojovou deskou přes sběrnici USB. Aplikace byla realizována v Matlabu (verze 2007a) a pro komunikaci s DSP využívá systém RTDX, popsaný v kapitole 3.3.5.

5.1 Grafické uživatelské rozhraní pro ovládání efektů

Grafické uživatelské rozhraní (GUI) programu pro ovládání efektů zobrazuje následující obrázek.



Obr. 5.1: Grafické uživatelské rozhraní ovládacího programu

V horní části se nachází tlačítka pro navázání komunikace s DSP, dále text informující o stavu připojení a tlačítko pro aktivaci LED indikátoru úrovně signálu. V dolní části se nachází panely s prvky pro ovládání jednotlivých efektů. Panely jsou seřazeny a očíslovány v pořadí, v jakém probíhá zpracování signálu v DSP. Tzn. jednotlivé algoritmy DSP jsou uspořádány sériově za sebou (viz kapitola 6), přičemž

pomocí tlačítka „*Aktivovat*“ lze zvolit, zda se daný efekt provede nebo vynechá. Nejprve se provádí regulace vstupní hlasitosti (panel č. 1), poté následují efekty zkreslení, dále delay efekty, pohyblivé filtry, stereo korektor a nakonec ekvalizér a nastavení výstupní hlasitosti. Většina parametrů je nastavitelná pomocí posuvníků v přednastavených hodnotách. Některé parametry je také možné nastavovat přímým zápisem číselné hodnoty do editačního pole, přičemž program hlídá, zda je vložené číslo v definovaném rozsahu. V rámci jednotlivých skupin efektů jsou také připraveny předvolby. Např. v panelu č. 3 lze rychle navolit efekt delay, flanger, chorus nebo vibráto. Navolené efekty je samozřejmě možné manuálně editovat.

Pro uložení stavu všech ovládacích prvků je v programu definováno statické pole (viz příloha C). V případě, že dojde ke změně nějakého ovládacího prvku, je tato nová hodnota uložena do tohoto pole a celé pole je následně odesláno pomocí rozhraní RTDX do signálového procesoru, kde dojde k aktualizaci proměnných ovládajících jednotlivé efekty. Pole pro přenos informací je celočíselného datového typu int16 (viz kapitola 5.2.2). Neceločíselné informace (jako zesílení) se před odesláním násobí zvoleným koeficientem, aby nedošlo ke ztrátě hodnoty vzniklé zaokrouhlením. Hodnoty, které jsou v decibelech, se před odesláním ještě přepočítají dle vztahu 4.36. Na straně DSP se přijaté hodnoty stejným koeficientem vydělí a uloží jako datový typ float.

Hodnoty jako zpoždění nebo hloubka modulace, které se nastavují v ms, se před odesláním přepočítávají přímo na celočíselný počet vzorků dle vztahu:

$$n = \frac{1000 \cdot d}{f_S}, \tag{4.37}$$

kde n je počet vzorků, d zpoždění v ms a f_S vzorkovací frekvence. Logické hodnoty, které informují např. o aktivaci / deaktivaci nějaké funkce, se přenáší v hodnotách 1/0. V rámci delay efektů je možné měnit i např. mezní frekvenci dolní propusti pro omezení pásma modulačního šumu (obr. 5.1: parametr „*Noise LPF*“). Přenášený údaj nese informaci pouze o mezní frekvenci filtru a přepočet koeficientů filtru probíhá až na straně signálového procesoru (dle vztahu 4.16).

Pomocí ovládacích prvků lze ovládat většinu parametrů jednotlivých efektů popsaných v kapitole 4. V rámci přehlednosti a rozumné velikosti okna GUI však nejsou zahrnuty některé méně důležité parametry. Do výsledného programu byly rovněž zařazeny pouze dva typy stereo korektorů, konkrétně varianty z obr. 4.11 a 4.13.

5.2 Implementace RTDX komunikace v Matlabu

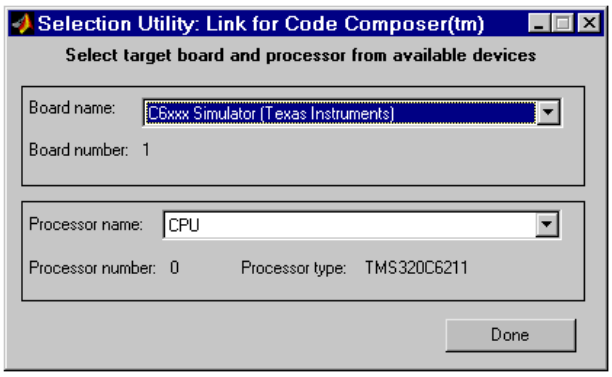
Pro podporu systému RTDX v Matlabu je dále potřebný toolbox *Link for Code Composer Studio*. Tento doplněk nabízí procedury pro konfiguraci, navázání a ukončení RTDX komunikace, dále příkazy pro diagnostiku a vlastní přenos dat. K dispozici je rovněž několik příkazů, pomocí kterých lze ovládat prostředí Code Composer Studio, které nalézají uplatnění při ladění a testování programů signálového procesoru. Z Matlabu lze zadat např. příkaz pro otevření požadovaného projektu v CCS (`open`), jeho následné nahrání do paměti DSP (`load`) a spuštění (`run`). Příkazem `visible` je možné zobrazit nebo skrýt okno prostředí CCS.

5.2.1 Inicializace RTDX komunikace

Inicializace RTDX v Matlabu sestává z několika kroků. Nejprve je nutné vybrat cílový HW, se kterým budeme komunikovat:

```
[boardNum,procNum] = boardprocsel;
```

Příkaz vyvolá dialog s volbou vývojové desky a procesoru zobrazený na obr 5.2. V případě, kdy máme připojenou pouze jednu desku s jediným procesorem, bude zobrazena pouze informace o cíli bez možnosti výběru. Po potvrzení získáme číslo připojené desky a procesoru.



Obr. 5.2: Dialog pro volbu cíle pro RTDX komunikaci

Oba získané údaje jsou potřebné pro vytvoření spojení (*link*) s prostředím CCS a systémem RTDX:

```
cc = ccsdsp('boardnum',boardNum,'procnum',procNum);
```

Link označený symboly *cc* využijeme pro další komunikaci Matlabu s prostředím CCS a signálovým procesorem.

Nyní přistoupíme k inicializaci RTDX kanálů, nejprve zvolíme velikost a počet vyrovnávacích bufferů:

```
configure(cc.rtdx,1024,1);
```

Velkost paměti by měla být v případě komunikace s 32 bitovými procesory minimálně o 8 bytů větší než velikost nejdelší přenášené zprávy. V případě, kdy budeme data pouze odesílat, postačí jeden vyrovnávací buffer. Poté vytvoříme kanál, který bude sloužit pro odesílání dat cílovému DSP:

```
open(cc.rtdx,'wchannel','w');
```

kde *wchannel* je symbolické označení kanálu a parametr *w* (*write*) určuje typ kanálu pro zápis (odesílání) dat. Kanál pro čtení dat by se vytvořil identickým příkazem s parametrem *r* (*read*).

Nyní aktivujeme rozhraní RTDX:

```
enable(cc.rtdx);
```

Vhodné je také nastavit časový limit (v jednotkách sekund), jak dlouho se bude program v Matlabu pokoušet o navázání spojení, aby nedošlo k trvalému zaseknutí programu v případě neúspěchu (výchozí hodnota činí 10 ms):

```
set(cc.rtdx,'timeout', 5);
```

Před aktivací kanálu pro zápis je již nutné, aby byl spuštěn program na DSP. Za předpokladu, že je program načten v paměti, otestujeme, zda program běží a pokud ne, spustíme ho příkazem `run`:

```
if (isrunning(cc)==0),
    run(cc);
end
```

Po spuštění je vhodné vyčkat krátký okamžik (1 s) a poté znovu otestovat, zda je program opravdu spuštěn a v negativním případě vypíšeme chybové hlášení:

```
pause(1)
if (isrunning(cc)==0)
    error('Chyba! Program v DSP není spuštěn.')
end
```

Na závěr inicializace se provede aktivace kanálu pro odesílání dat:

```
enable(cc.rtdx, 'wchannel');
```

A opět je vhodné ověřit, zda se tento krok provedl a v případě, že ne, upozornit uživatele chybovým hlášením.

```
if (isenabled(cc.rtdx, 'wchannel'))
    error('Chyba! RTDX kanál nebyl vytvořen.')
```

```
end
```

5.2.2 Odesílání dat RTDX kanálem

Následuje ukázka skriptu v Matlabu, který odešle pole dat cílovému DSP pomocí kanálu RTDX. Celý kód je vhodné vložit do bloku `try–catch`, který zabrání nechtěnému zastavení celé aplikace při případné chybě. Před odesláním dat se nejprve testuje, zda lze do přidělené vyrovnávací paměti zapisovat.

```
stateData = [ones(1,50)];
try
    if iswritable(cc.rtdx,'wchannel'),
        writemsg(cc.rtdx,'wchannel', int16(stateData));
    end
catch
end
```

Odesílat lze zprávy obsahující pole dat. Podporovány jsou datové typy *uint8*, *int16*, *int32*, *single* a *double* a maximální délka pole je omezena velikostí vyrovnávacího bufferu RTDX kanálu. Datový typ single má v Matlabu rozsah pouze -1 až 1, proto jej nelze na straně DSP přijímat jako datový typ float. Pro přenos byl proto použit úspornější celočíselný datový typ int16. Rozsah tohoto typu (-32,768 až 32,767) je však pro přenos ovládacích informací dostatečný.

Pro přenos ovládacích povelů je použito pole *stateData* o padesáti prvcích. Význam veškerých přenášených dat je uveden v příloze C. Např. první položce pole odpovídá parametr nastavení vstupní hlasitosti.

Veškeré potřebné informace pro implementaci RTDX byly čerpány z dokumentace Matlabu [13].

5.3 Implementace RTDX komunikace na DSP

Pro komunikaci mezi Matlabem a deskou s DSP je nutné implementovat RTDX i na straně signálového procesoru. Následuje postupný popis programu v jazyce C, který čte data odeslaná z Matlabu a přijímá je ze vstupního kanálu RTDX:

```
//RTDX.c
#include <rtdx.h>                 // podpora RTDX komunikace
#include "target.h"               // inicializace RTDX cíle
short recvd[50];                  // pole pro příjem zprávy
RTDX_CreateInputChannel(ichan);   // kanál pro příjem dat
RTDX_CreateOutputChannel(ochan); // kanál pro odesílání dat
```

Nejprve je nutné připojit do programu potřebné knihovny funkcí. Součástí standardní instalace prostředí CCS je knihovna pro podporu RTDX. Deklarace těchto funkcí jsou umístěny v hlavičkovém souboru *rtdx.h*. Pro inicializaci RTDX komunikace je dále potřeba soubor *target.h*. Následuje deklarace pole pro příjem zprávy a datových struktur realizující vstupní a výstupní RTDX kanál.

Veškeré zpracování signálu běží v rámci obsluhy přerušení, které se vyvolá pokaždé při přijetí nového vzorku AD převodníkem:

```
// obsluha přerušení
void c_int11(){
     ...           // zpracování signálu, algoritmy audio efektů
}
```

Následuje hlavní smyčka programu:

```
// hlavní smyčka
void main( void ){
     TARGET_INITIALIZE();    // inicializace RTDX
     while(1) {
                             // test příchodu zprávy
          while (!RTDX_isInputEnabled(&ichan)) {}
                             // načte zprávu do pole recvd
               RTDX_read( &ichan, recvd, sizeof(recvd));
               input_volume = recvd[0];
               ...
               }
}
```

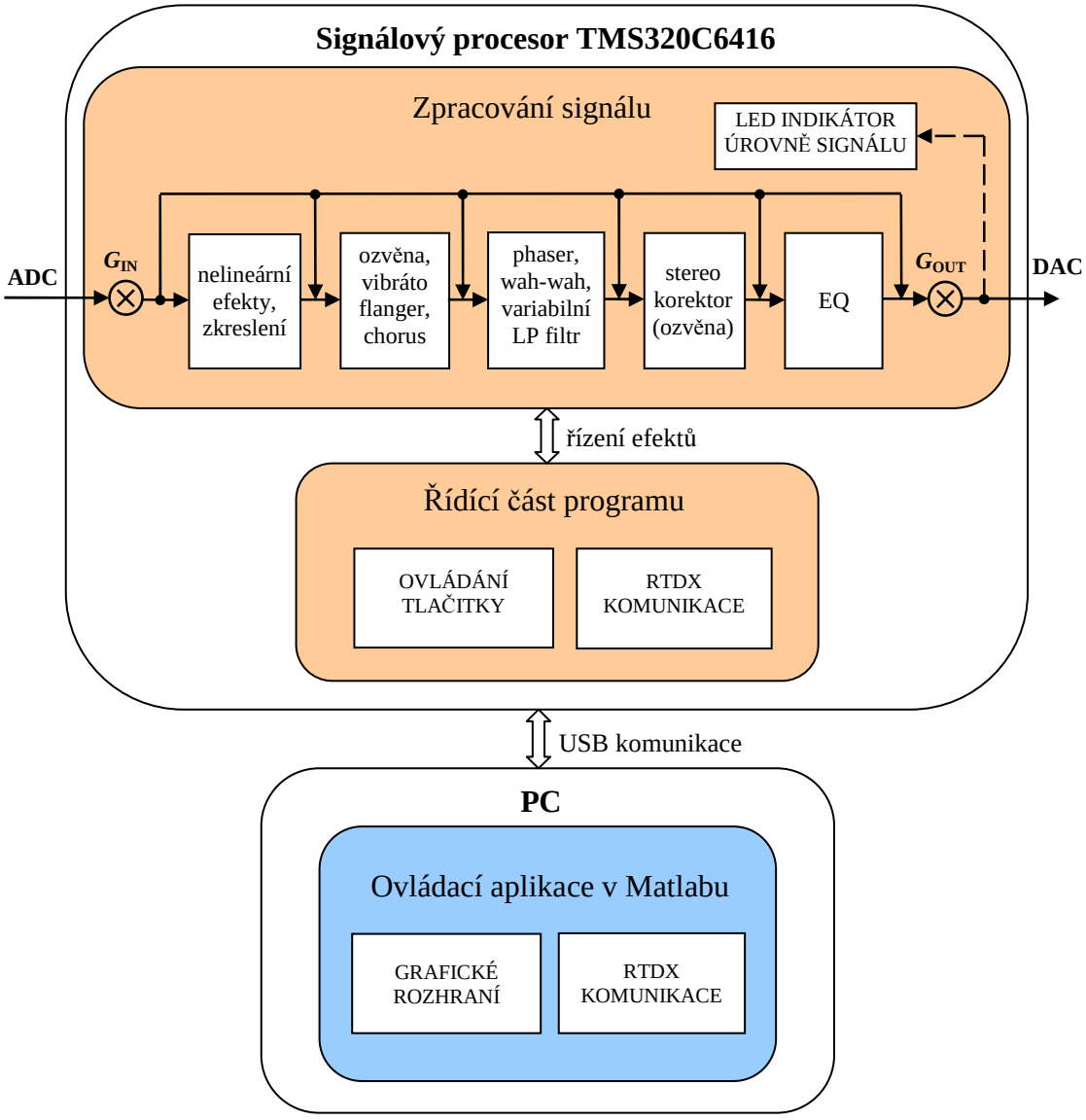
V hlavní smyčce se po spuštění programu provede inicializace RTDX případně další operace, které je nutné provést po restartu DPS. Poté následuje nekonečná smyčka, ve které program neustále testuje, zda nepřišla nová zprava. Po příchodu zprávy se data přečtou a uloží do připraveného pole. Hodnoty v tomto poli je již možné používat pro ovládání efektů. Např. na pozici pole *recvd*[0] je poslán údaj o vstupním zesílení (indexování polí v C začíná od 0, kdežto v Matlabu od 1).

Příklady implementace RTDX komunikace v DSP lze nalézt v [2], [13].

6 Celkové schéma programu realizovaných efektů

Výsledné blokové schéma realizovaných efektů je rozděleno na program signálového procesoru a počítačovou aplikaci realizující ovládací rozhraní. V programu DSP vidíme tok zpracovávaného signálu a bloky reprezentující algoritmy jednotlivých efektů doplněné o regulaci vstupní a výstupní hlasitosti a indikátor úrovně signálu. V procesoru jsou rovněž prováděny veškeré přepočty koeficientů variabilních filtrů.

Řídící část programu zahrnuje ovládání pomocí tlačítek a komunikaci s hostitelským PC pomocí systému RTDX. Počítačová aplikace v Matlabu je tvořena ovládacím grafickým rozhraním a dále programem, který realizuje RTDX komunikaci s DSP.



Obr. 6.1: Celkové blokové schéma realizovaných digitálních audio efektů

7 Zhodnocení výsledků a závěr

Účelem této práce bylo nastudovat principy základních typů digitálních audio efektů a jejich realizace s využitím přípravku osazeného digitálním signálovým procesorem TMS320C6416. Některé vytvořené efekty lze z hlediska subjektivního posouzení zvukové kvality označit za více, jiné za méně zdařilé. Efekty realizované pomocí číslicových filtrů s konstantními i variabilními koeficienty (ekvalizér, wah-wah, phaser) zní zvukově přirozeně a čistě. Dá se říci, že plně využívají možnosti použitého hardwaru a příliš se neliší od efektů, které nabízí např. běžně dostupné programy pro editaci audio nahrávek. Rovněž efekty realizované pomocí časových modulací (vibrato, chorus, flanger) nabízí při vhodném nastavení parametrů rozmanité a zajímavé výsledky srovnatelné např. s programy pro úpravu zvuku dodávanými k běžným zvukovým kartám. Avšak najít optimální nastavení parametrů algoritmu realizujícího tyto efekty není úplně snadné a některé kombinace tak vedou k méně vydařeným výsledkům. Rovněž se ukázalo, že algoritmy používané pro časově modulované efekty jsou při určitém nastavení poměrně náchylné na šum obsažený ve vstupním signálu, a v určitých případech dochází ke znatelnému zvýraznění těchto rušivých složek. V takovém případě jsou kladeny vyšší nároky na kvalitu připojeného zdroje audio signálu a propojovacího kabelu. Jako naprosto nevyhovující se ukázal audio výstup PC vyvedený na konektorech z přední strany skříně běžně dodávaným (nedostatečně stíněným) kabelem. Šum indukovaný v signálovém kabelu uvnitř počítače se negativně projeví po aktivaci některých efektů zejména při poslechu pomocí sluchátek.

Věnovali jsme se také zkreslujícím efektům pro elektrickou kytaru. Ačkoliv se na první pohled zdá, že podobná úloha, kdy se snažíme záměrně zkreslit kytarový signál, je snadná, ukazuje se, že se jedná o rozsáhlejší problematiku. Realizované algoritmy popsané v kapitole 4.4 nabízí poměrně zdařilé kytarové efekty pro některé tóny a akordy, avšak pro jiné již nezní příliš dobře. Tento efekt tak nelze srovnávat s možnostmi, které nabízí výrobky renomovaných světových společností zabývajících se vývojem a výrobou kytarových efektů. Pro dosažení lepších výsledků by bylo nutné použít pokročilejší techniky, například simulace analogových kytarových efektů na základě aproximace diferenciálních rovnic, kterými se zabývá článek [14].

Naopak za zdařilé lze označit také efekty pro tvorbu dozvuku případně vytváření stereo vjemu u mono signálů popsaných v kapitole 4.2. Ačkoliv se nejedná o složité algoritmy, zvukově nabízí poměrně zajímavé výsledky.

Možné vylepšení z hlediska výpočetní náročnosti realizovaných efektů by mohlo spočívat např. v pokročilejší optimalizaci algoritmů DSP v assembleru s využitím technik paralelního programování. Avšak pro dostatečný výkon DSP nebyla další optimalizace nutná a vzhledem k tomu, že použitý signálový procesor nepodporuje HW výpočty v plovoucí řádové čárce, by byla případná implementace neceločíselných výpočtů v assembleru i značně složitá.

První část práce tedy splňuje požadavky zadání. Zvukové ukázky vybraných audio efektů lze nalézt na přiloženém CD ve formátu wav. Na realizované efekty by se dalo navázat využitím dalších technik pro zpracování signálu. Zavést lze blokového zpracování signálu doplněné metodou sčítání přesahů (OLA) a výpočtem FFT, IFFT a následné zpracování signálu ve spektru. Pomocí výpočtů ve spektru lze realizovat například pokročilejší simulaci dozvuku poslechového prostoru (*hall*), dále efekty pro změnu výšky tónu (*pitch shifting*), případně efekt *oktáver*, který doplňuje zahraný tón o variantu posunutou o oktávu níže, případně výše [1]. Nevýhodou užití blokového zpracování je naopak zavedení programového zpoždění zpracovávaného signálu.

Jako druhý hlavní úkol lze označit tvorbu programu pro ovládání jednotlivých efektů, který nabízí daleko lepší možnosti než-li ovládání pomocí několika tlačítek umístěných na vývojové desce se signálovým procesorem. Pro komunikaci mezi DSP a PC byl využit systém RTDX, který slouží zejména pro ladění programu DSP pomocí počítačových programů v reálném čase, ale lze jej použít i pro ovládání signálového procesoru. Zde se vyskytly problémy zejména s kompatibilitou jednotlivých softwarových prostředků. Ovládací program na straně PC byl nakonec realizován v Matlabu (verze 2007a). Avšak ukázalo se, že toto prostředí nabízí dostatečné možnosti i co se týče tvorby grafických aplikací pro podobné účely, a výsledná aplikace tak umožňuje pohodlné ovládání jednotlivých efektů.

Seznam použité literatury

[1] Zölzer U. et al.: DAFX: Digital Audio Effects, Wiley, ISBN-13: 978-0471490784, 2002.

[2] Chassaing R.: Digital Signal Processing and Applications with the TMS320C6713 and TMS320C6416 DSK, Wiley-IEEE Press, ISBN-13: 978-0470138663, 2008.

[3] Davídek, V., Sovka, P.: Číslicové zpracování signálů a implementace, 1. vydání, Vydavatelství ČVUT , Praha 1996, 80-01-01530-0.

[4] Texas Instruments, TMS320C6000 Code Composer Studio Tutorial, revision C, February 2000 [online]. [cit. 2011-04-25]. Dostupný z WWW: http://focus.ti.com/lit/ug/spru301c/spru301c.pdf

[5] SPECTRUM DIGITAL, INC., TMS320C6416 DSK Technical Reference, Rev. A, April 2003, [online]. [cit. 2011-04-25]. Dostupný z WWW: http://c6000.spectrumdigital.com/dsk6416/V1/docs/dsk6416_TechRef.pdf

[6] Sýkora R., Krutílek F., Včelař J.: Elektronické hudební nástroje a jejich obvody, 1. vydání, SNTL – Nakladatelství technické literatury, Praha 1981, L26-B2-IV-31/52294

[7] Smetana C. a kolektiv: Praktická elektroakustika, vydání 1., Nakladatelství SNTL, Praha 1981, L 26-E1-IV-41/52322

[8] Smékal Z., Sysel P.: Signálové procesory, 1. vydání, Nakladatelství Sdělovací technika, Praha 2006, ISBN: 80-86645-08-8

[9] Kraus A.: Obvod pro řízení stereofonní báze, Stavebnice a konstrukce A Radio, ročník IV/2000 číslo 3, Praha 2000, s. 3 – 5, ISSN 1212-1843

[10] Texas Instruments, TMS320C64x Technical Overview, January 2001 [online]. [cit. 2011-04-25]. Dostupný z WWW: http://focus.ti.com.cn/cn/lit/ug/spru395b/spru395b.pdf

[11] Texas Instruments, TMS320C6414T, TMS320C6415T, TMS320C6416T Fixed-Point Digital Signal Processors Data Sheet, November 2003 – Revised April 2009 [online]. [cit. 2011-04-25]. Dostupný z WWW: http://focus.ti.com/lit/ds/symlink/tms320c6416t.pdf

[12] Texas Instruments, TMS320C6414T/15T/16T Power Consumption Summary, February 2008 [online]. [cit. 2011-04-25]. Dostupný z WWW: http://focus.ti.com/lit/an/spraa45a/spraa45a.pdf

[13] The MathWorks, Inc., Dokumentace prostředí Matlab, verze 2007a

[14] Macák J., Shimmel J., Real-Time Guitar Tube Amplifier Simulation using an Approximation of Differential Equations, Proc. of the 13th Int. Conference on Digital Audio Effects (DAFx-10), Graz September 2010 [online]. [cit. 2011-04-25]. Dostupný z WWW: http://dafx10.iem.at/proceedings/papers/MacakSchimmel_DAFx10_P12.pdf

Příloha A – Popis obsahu přiloženého CD

- Elektronická verze textu práce ve formátu pdf a doc
- Obrázky a grafy použité v textu – formáty jpg, png a fig (Matlab)
- Zdrojové kódy programů Matlab a C (Code Composer Studio)
- Zvukové ukázky vybraných audio efektů ve formátu wav

Příloha B – Ukázka zdrojového kódu vybraných efektů

Hlavní část zdrojového kódu jednotky pro časově modulované efekty v C:

```
// Universal Comb Filter
...
if(MOD_MODE==1){                                    // modulace šumem
   x = 0.0000610351*((rand()) - 16384);             // generování šumu
   // filtrace x LP filtrem a výpočet aktuální neceločíselné hodnoty zpoždění:
   actDelay_L = width + delay + width * filter(x);
}
else if(MOD_MODE==2){                               // modulace šumem + sinem
   x = 0.0000610351*((rand()) - 16384);             // generování šumu
   // výpočet aktuální neceločíselné hodnoty zpoždění:
   actDelay_L = width + delay_width + delay + delay_width*
   sin(2*pi*fm*((float)time)/((float)Fs)) + width * filter(x);
}
else{                                               //modulace sinem
      // výpočet aktuální neceločíselné hodnoty zpoždění:
      actDelay_L = width + delay + width * sin(2*pi*fm*((float)time)/((float)Fs));
}
M = ceil(actDelay_L);                          // horní celá část actDelay
Mfrac = actDelay_L + 1 - M;                    // desetinná část actDelay
index1 = (delaylength-M+ii)%delaylength;       // index vzorku xh zpožděného o M
index2 = (delaylength-M+ii+1)%delaylength;     // index vzorku xh zpožděného o M-1
//lineární aproximace dvou zpožděných vzorků:
linAprox_L = buffer_L[index1]*Mfrac +(buffer_L[index2])*(1-Mfrac);
xh = inputL*gain + FB*linAprox_L;              //stavová proměna zv. struktury
inputL = FF*linAprox_L + BL*xh;                //vystup sloučen
// uloží stavovou proměnnou Xh zpožďovacího bufferu:
buffer_L[ii] = xh;
...
```

Zdrojový kód efektu Wah-wah v Matlabu:

```
% Auto WAH-WAH (modulace)

x = wavread('record.wav');% načtení zvukových dat
y = zeros(1,length(x));   % příprava dat. polí
ya = zeros(1,length(x));  % příprava dat. polí
fb=300                    % spodní mezní frekvence BP filtru
fc=1000                   % horní mezní frekvence BP filtru
fcfb = fc/fb              % 1-5
% inicializace koeficientu filtru:
c = (tan(pi*fb/fs) -1) / (tan(2*pi*fb/fs)+1);
d =  -cos(2*pi*fc/fs);
fmin = 100                % min rozsah ladění filtru
fmax = 1500               % max rozsah ladění filtru
rate = 3                  % modulační frekvence
band = 0.9/2              % relativní šířka pásma filtru
force =0.6                % intenzita efektu 1... 100%

% filtrace přeladitelným PB filtrem realizovaný z allpass filtru
for n=3:length(x)
    fc = fmin + 0.5*fmax + 0.5*fmax*(sin(2*pi*rate*n/fs));     % přepočet fc
    fb =fc*band;                                               % přepočet fb
% přepočet koeficientů filtru:
    c = (tan(pi*fb/fs) -1) / (tan(2*pi*fb/fs)+1);
    d =  -cos(2*pi*fc/fs);
% filtrace:
    ya(n) = -c*x(n) + d*(1-c)*x(n-1) + x(n-2) -d*(1-c)*ya(n-1) + c*ya(n - 2);
    y(n) = x(n) - force*ya(n);
end
wavwrite(y,Fs,'output.wav')
```

Příloha C – Význam dat při komunikaci mezi PC a DSP

Přehled dat odesílaných z PC programu (Matlab) do programu DSP (C) RTDX kanálem. Pro odesílání slouží pole *stateData*[*index*] datového typu int16. Tabulka shrnuje význam jednotlivých položek pole (indexovaných dle Matlabu (od hodnoty 1):

Index	Panel GUI	Význam hodnoty	Proměnné editačních prvků v Matlab GUI	Symbol proměnné v C (DSP)	Výchozí hodn. v GUI	Koef.	Min	Max
Ovládání hlasitosti a LED indikátoru:								
1	1	vstupní hlasitost (dB)	volume_in_slider volume_in_edit	volume_in	0	1000	-40	20
2	6	výstupní hlasitost (dB)	volume_out_slider volume_out_edit	volume_out	1	1000	-40	20
3	–	balance (L/R)	*nevyužito*	rvol, lvol	0.5	1000	0	1
4	horní	aktivace LED indikátoru	led_checkbox	vu_meter	1	1	0	1
Ovládání efektů zkreslení:								
5	2	aktivace zkreslení	dist_checkbox	distortion_state	0	1	0	1
6	2	typ zkreslení (hard/soft)	dist_hard_ checkbox dist_soft_ checkbox	dist_type	1/ 0	1	0	1
7	2	boost zkreslovače	dist_boost_slider	boost	10	1000	0	20
8	2	ekvalizér – neomezené pásmo	dist_all_slider	dist_all	0	1000	-3	3
9	2	ekvalizér LP	dist_lp_slider	dist_lp	1	1000	-3	3
10	2	ekvalizér BP	dist_bp_slider	dist_bp	1	1000	-3	3
11	2	ekvalizér HP	dist_hp_slider	dist_hp	1	1000	-3	3
12	2	LF pásmo LP1	dist_lp1_slider	dist_lp1	0	1000	0	3
13	2	LF pásmo LP2	dist_lp2_slider	dist_lp2	0	1000	0	3
14	2	LF pásmo LP3	dist_lp3_slider	dist_lp3	0	1000	0	3
15	2	LF pásmo LP4	dist_lp4_slider	dist_lp4	0	1000	0	3
Ovládání časově modulovaných efektů:								
16	3	aktivace delay efektů	ucf_state_checkbox	ucf_state	1	1	0	1
17	3	zesílení delay efektů	ucf_gain_slider	ucf_gain	0,9	1000	0	1
18	3	zpoždění (ms)	ucf_delay_slider ucf_delay_edit	ucf_delay ucf_delay	0	fs/1000	0	320
19	3	typ modulace (1: sin, 2: noise, 3: noise+sin)	ucf_mod_type	ucf_mod_type	1	1	1	3
20	3	hloubka modulace (procenta z hodn. zpoždění)	ucf_width_slider	ucf_width	0	0.01*fs	0	100
21	3	modulační frekvence delay efektů (Hz)	ucf_freq_slider	ucf_freq	0	1000	0	10
22	3	aktivace stereo fáze (-1: on, 1: off)	ucf_stereo_checkbox	ucf_stereo	1	1	-1	1
23	3	bypass vazba (parametr *BL*)	ucf_bl_slider	ucf_bl	1	1000	-1	1
24	3	dopředná vazba (parametr *FF*)	ucf_ff_slider	ucf_ff	0	1000	-1	1
25	3	zpětná vazba (parametr *FB*)	ucf_fb_slider	ucf_fb	0	1000	-0,9	0,9
41	3	mezní frekvence LP filtru šumu	ucf_noiselpf_slider	ucf_noiselpf	50	1	10	1010
Ovládání efektů realizovaných pohyblivými filtry:								
26	4	aktivace efektu s variabilními filtry	spectral_checkbox	spectral_state	0	1	0	1
27	4	Volba: 1: wah-wah, 2: phaser, 3: variabilní LP	spectral_popupmenu	spectral_type	1	1	1	3
28	4	modulační frekvence pohyblivých filtrů (Hz)	spectral_fmod_edit spectral_fmod_slider	spectral_fmod	1	1000	0.1	10
29	4	horní mez ladění filtru (Hz)	spectral_fmax_edit	spectral_fmax	1500	1	20	20k
30	4	spodní mez ladění filtru (Hz)	spectral_fmin_edit	spectral_fmin	200	1	20	20k
Ovládání stereo korektoru:								
31	5	aktivace stereo korektoru	enh_checkbox	enh_state	1	1	0	1
32	5	stereo efekt pro mono signál	enh_mono	enh_type	1	1	0	1
33	5	rozšířeni stereo efektu	enh_stereo	enh_type	0	1	0	1
34	5	intenzita stereo efektu	enh_depth_edit enh_depth_slider	enh_depth	5	1	0	30
35	5	zpoždění stereo korektoru	enh_delay_edit enh_delay_slider	enh_buff	50	fs/1000	0	320
Ovládání ekvalizéru:								
36	6	aktivace ekvalizéru	eq_checkbox	eq_state	1	1	0	1
37	6	basy (dB)	eq_lp_slider	eq_lp	0	1000	-12	12
38	6	středy (dB)	eq_bp1_slider	eq_bp1	0	1000	-12	12
39	6	vyšší středy (dB)	eq_bp2_slider	eq_bp2	0	1000	-12	12
40	6	výšky (dB)	eq_hp_slider	eq_hp	0	1000	-12	12
42-50		*volné pozice – případné rozšíření*						

Pozn. 1: fs je vzorkovací frekvence (výchozí hodnota 48 kHz)

Pozn. 2: Indexování polí v C začíná od 0, platí *index* -1

Pozn. 3: Koeficientem se násobí desetinná hodnota v Matlabu před odesláním v celočíselném formátu, stejnou hodnotou se na straně DSP proměnná vydělí a dále používá jako datový typ float.